Computational Aerosciences Branch Summer Intern Presentations

1. ***Database Support for the SLS Artemis-2 Booster Separation*** by **Allen Ruan**, UC Berkeley, CA

2. ***One Dimensional Point Distribution using Geometric Stretching*** by **Sam Aslam**, Wesleyan University, CT

3. ***Development of a Triangulation Quality Checker for LAVA*** by **Jacob Zenger**, University of Utah, UT

4. ***Development of a Utility to Automatically Generate Trajectory Files Based on User Prescribed Motions*** by **Keshav Sriram**, Diamond Bar High School, CA

5. ***Development of a Mesh Redistribution Algorithm for Structured Curvilinear Meshes*** by **Chase Ashby**, University of Kentucky, KY

6. ***Uncertainty Quantification with Cart3D and QUEST*** by **Liam Smith**, Georgia Tech, GA

7. ***Unstructured CFD support for Commercial Supersonic Technology Wind Tunnel Tests*** by **Robert Comstock** (Cal Poly San Luis Obispo, CA) and **William Bowes** (UC Davis, CA)

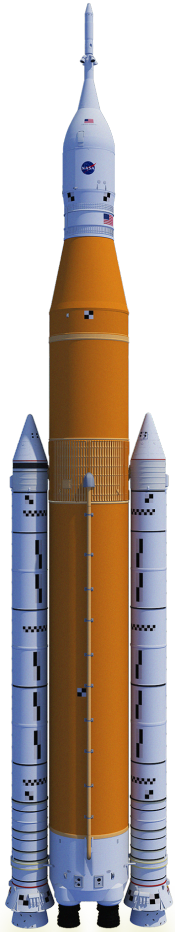
Advanced Modeling & Simulation Seminar Series,
NASA Ames Research Center July 30, 2019

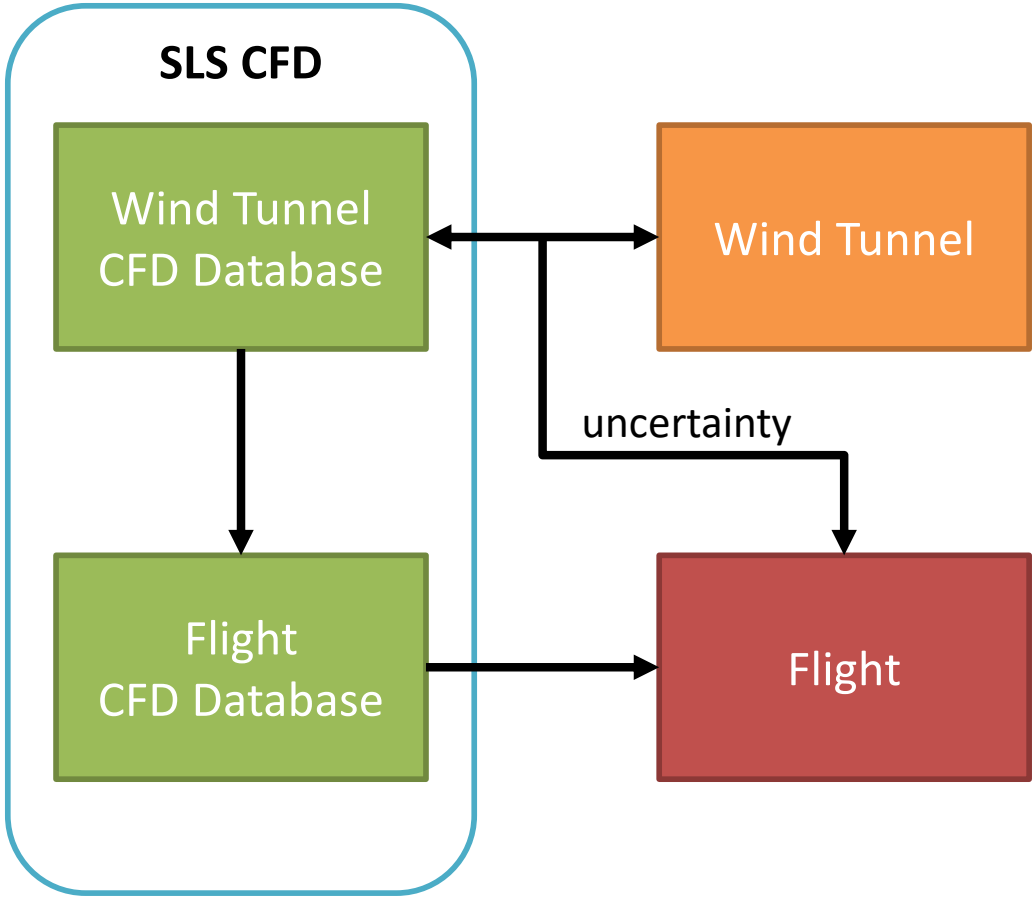
Database Support for the SLS Artemis 2 Booster Separation

Allen Ruan
UC Berkeley, CA

July 30, 2019

Motivation



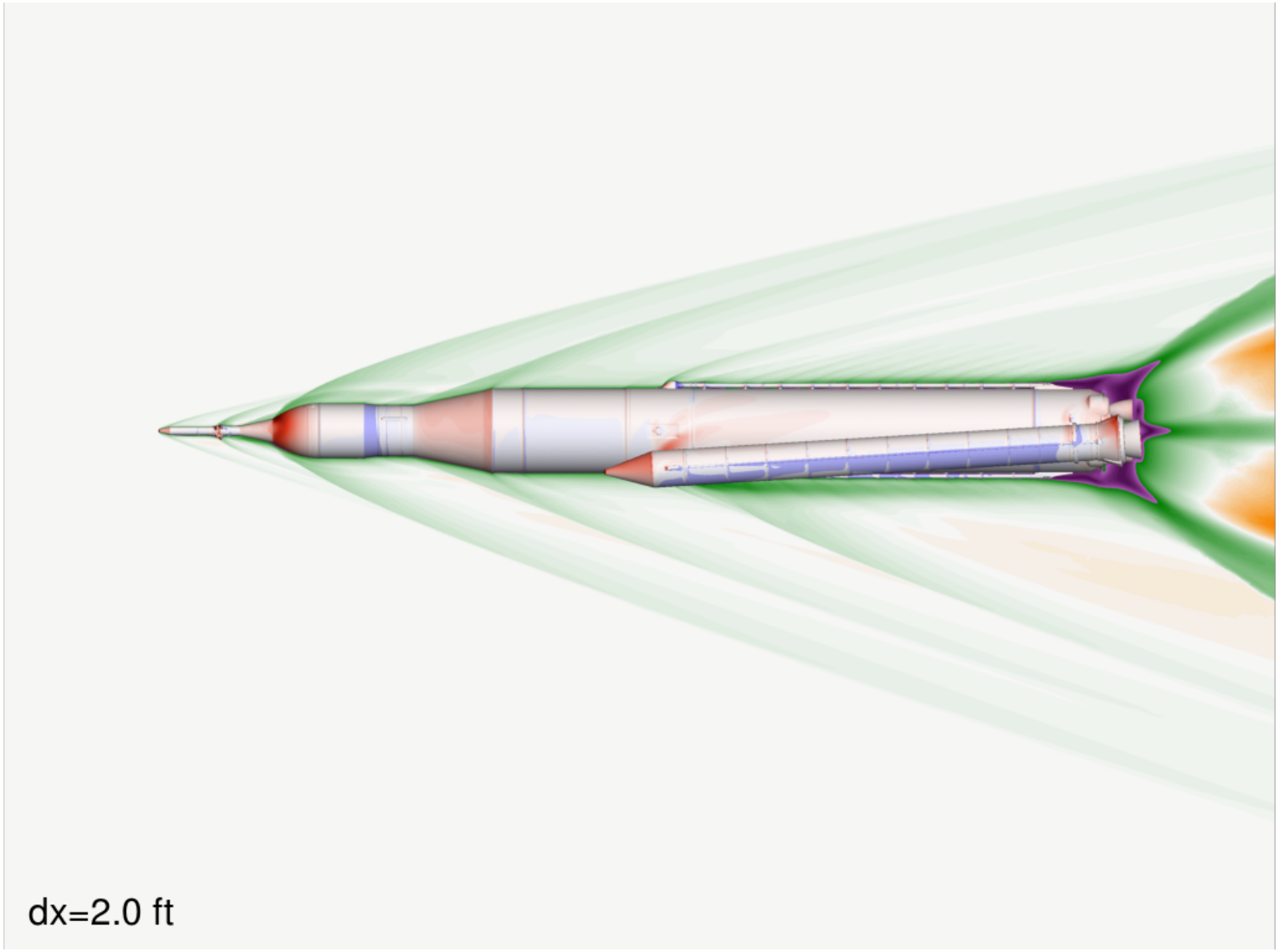


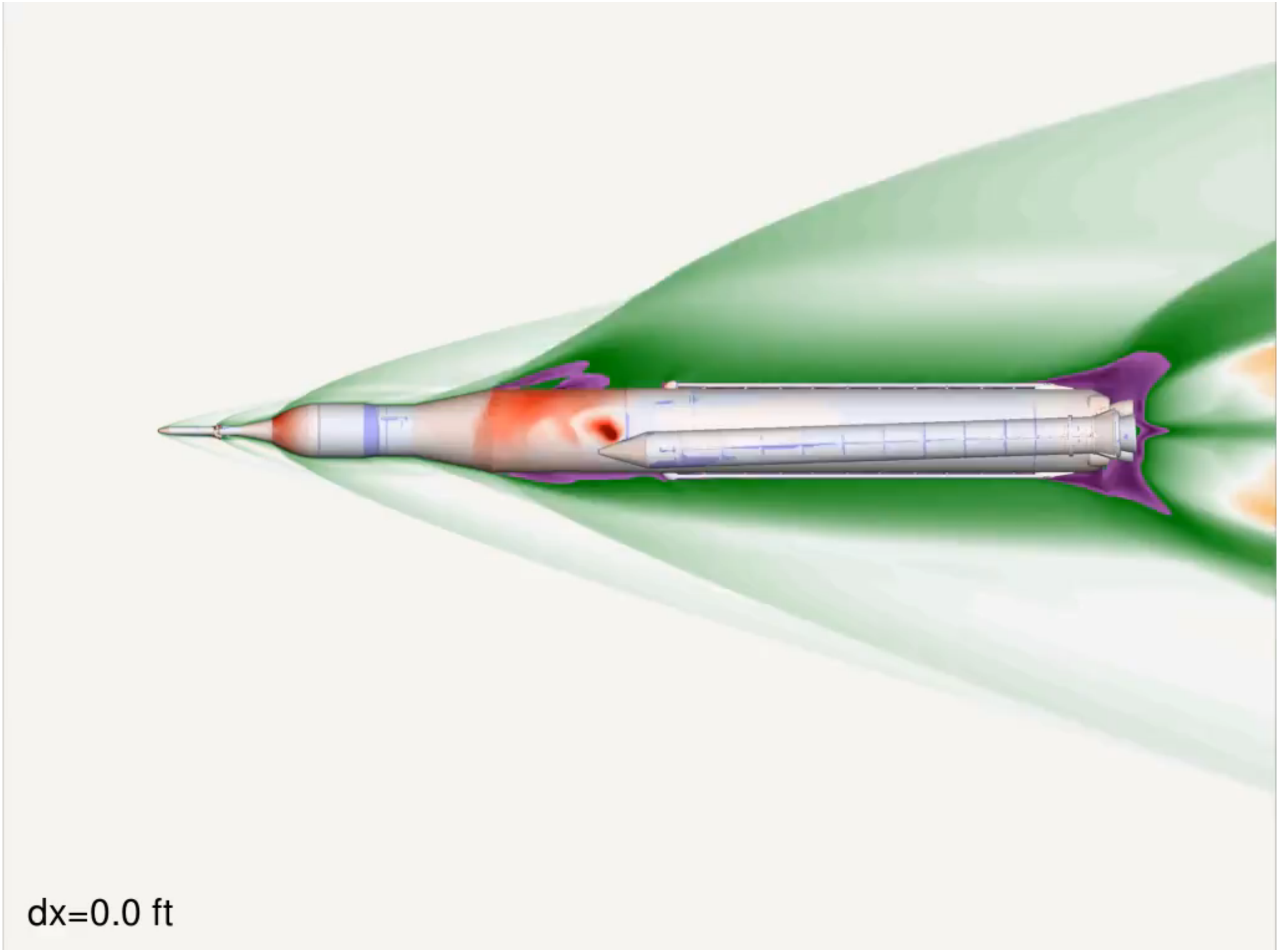
Utilized CAPE tools to run and post-process CFD simulations on NAS Supercomputers to support the development of the booster separation database for SLS Artemis 2

STS-117 SRB Separation Camera EHV217

Marshall Space Flight Center
Engineering Photographic Analysis

Post-Processing




dx=0.0 ft

Individual Stats

	Wind Tunnel		Flight		Total
	BSM on	BSM off	Nominal	CSE 1 Out	
Cases Run	103	504	1760	1050	3417
Approx. Core Hours					9.3 million

Reflections

Learning/Growth

- How to read/interpret CFD plots
- Introduced to the many components of creating a database including writing modules in Python
- Utilizing tecplot, overgrid, and pyfun
- Interfacing with BASH terminals, vim, and supercomputers (pfe/lfe)
- Documentation of reports using Sphinx
- Adapting to SLS CFD team

Future Improvement

- Write more scripts to further automate/optimize certain processes to facilitate creating and analyzing databases.
- Continued/Expanded rigorous documentation of processes, errors, successes

Acknowledgements

- Dr. Cetin Kiris
- SLS CFD Team – Dr. Stuart Rogers, Dr. Derek Dalle, Henry Lee, Jamie Meeroff
- Interns

Thank you

1-D Point Distribution using Geometric Stretching with Uniform Padding

Samuel Aslam
Wesleyan University, CT

July 30 2019

Introduction

ABSTRACT Extending geometric stretching methodology that allows for spacing to be specified from one or both ends. Taking maximum stretching ratio and maximum spacing as input variables the goal is to have a point distribution function that satisfies the given constraints by potentially using the least amount of points.

BIO Samuel Aslam. Rising senior at Wesleyan University pursuing a B.A in Mathematics.

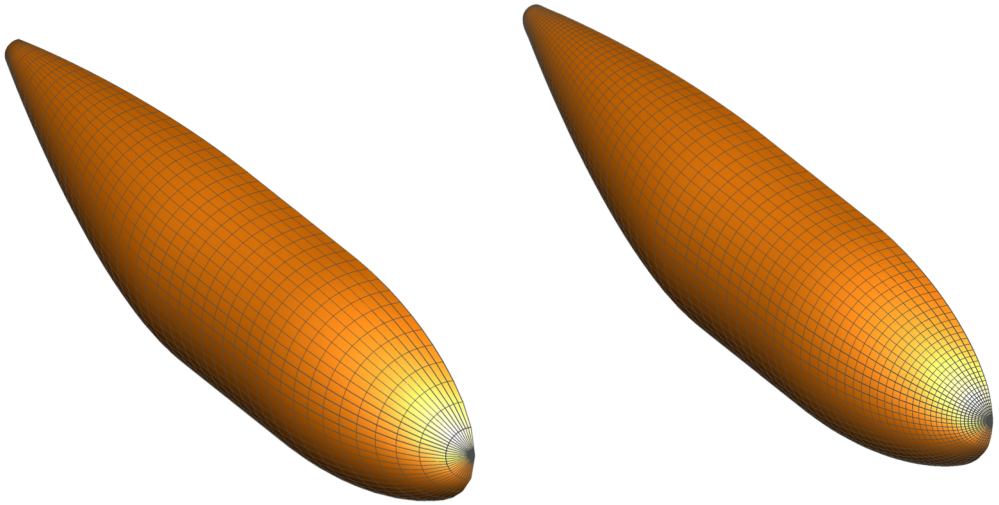
MENTORS William. M. Chan, Shishir Pandya

Objective

- A grid is discrete representation of a geometric object and it should have the appropriate point distribution to resolve the geometry and the flow physics.

- **Hyperbolic Tangent Stretching:**
  - Inputs:
    - Domain size
    - Number of points
    - Spacing at one or both end
  - Varying stretching ratio

- **Geometric Stretching**:
  - Inputs:
    - Domain size
    - Number of points
    - Spacing at one end only
  - Constant stretching ratio



- **Typical usage:**
  - specify spacing at one or both ends
  - maximum spacing
  - maximum stretching ratio

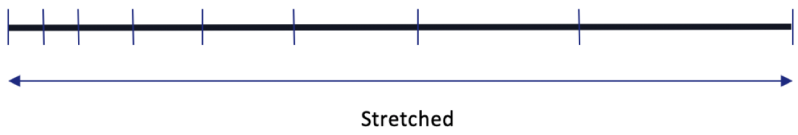
Objective:
Develop a python function to explore geometric stretching with typical usage constraints

One-Sided Geometric Stretching with Uniform Padding

- Given
  - domain D
  - initial spacing Δs_i
  - maximum spacing Δs_{max}
  - maximum stretching ratio r_{max}

- Find a point distribution that satisfies the constraints with as few points as possible.
- Case 1 (stretched region only, largest $\Delta s < \Delta s_{max}$)

$$\Delta s_i \left(\frac{1 - r^{\frac{ln(\frac{\Delta s_{max}}{\Delta s_i})}{ln(r)}}}{1 - r} \right) \geq D$$



- Case 2 (stretched + uniform region)

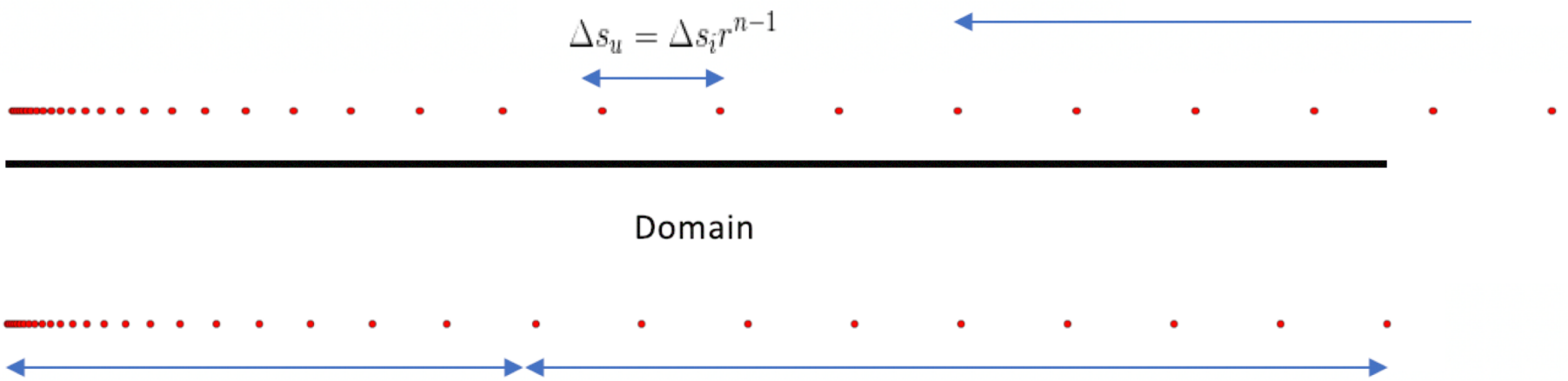
$$\Delta s_i \left(\frac{1 - r^{\frac{ln(\frac{\Delta s_{max}}{\Delta s_i})}{ln(r)}}}{1 - r} \right) < D$$

One-Sided Case 2

Stretched Region + Uniform Region

1) Build point distribution from Δs_i, stretch using r_{max} until reaches Δs_{max}
2) Pad remaining domain with uniform spacing = last spacing from stretched region (Δs_u)
3) Calculate m = number of uniform spacings
4) n = number of points from Δs_i side
5) Solve equation for r

$$\Delta s_i \left(\frac{1-r^n}{1-r}\right) + m\Delta s_u = D$$



Two-Sided Geometric Stretching with Uniform Padding

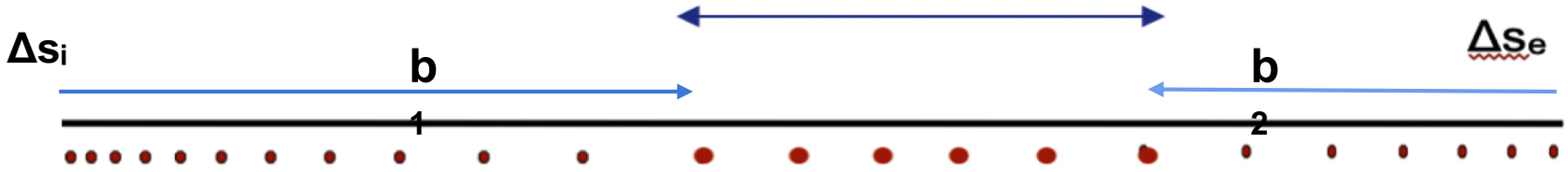
- Given
  - domain D
  - initial spacing Δs_i
  - end spacing Δs_e
  - maximum spacing Δs_{max}
  - maximum stretching ratio r_{max}
- Find a point distribution that satisfies the constraints with as few points as possible.
- **Case 1**
  - $b_1 + b_2 > D$
  - Δs does not hit Δs_{max} from either side
- **Case 2**
  - $b_1 + b_2 > D$
  - Series hit Δs_{max} from one side only
- **Case 3**
  - $b_1 + b_2 < D$
  - Series hit Δs_{max} from both sides

Distance reached by stretching at r_{max} from Δs_i

$$b_1 = \Delta s_i \left(\frac{1 - r^n}{1 - r}\right)$$

Distance reached by stretching at r_{max} from Δs_e

$$b_2 = \Delta s_e \left(\frac{1 - r^m}{1 - r}\right)$$



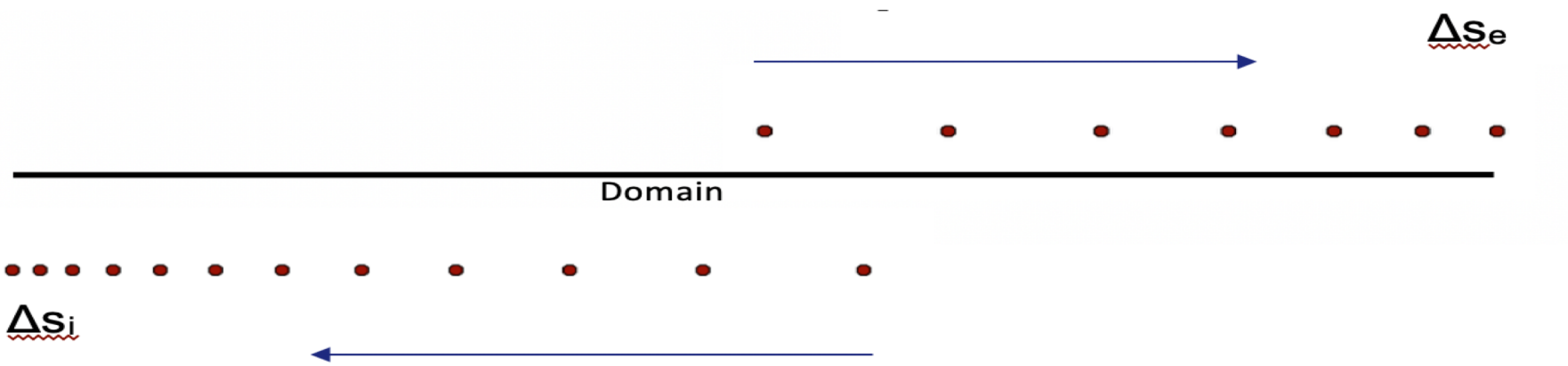
Two-Sided Case 1

1) Build point distribution from Δs_i and Δs_e, stretch using r_{max} but Δs_{max} never reached. Stop when sum of both sides ($b_1 + b_2$) exceeds domain D
2) n = number of points from Δs_i side

   m = number of points from Δs_e side

1) Solve equation for r

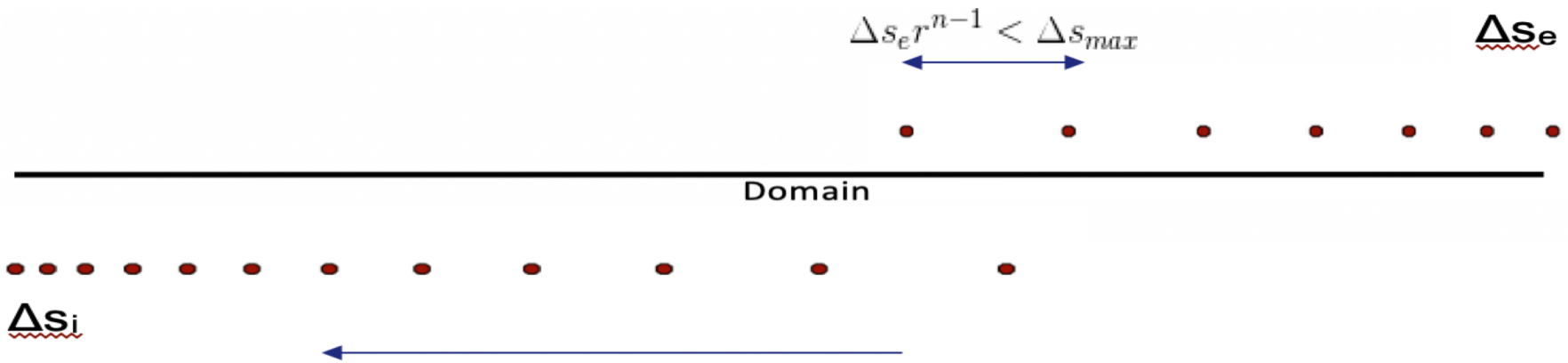
$$\Delta s_i\left(\frac{1-r^n}{1-r}\right) + \Delta s_e\left(\frac{1-r^m}{1-r}\right) = D$$



Two-Sided Case 2

1) Build point distribution from Δs_i and Δs_e, stretch using r_{max} until Δs_{max} reached from one side only
2) Build point distribution from other side until sum of both sides ($b_1 + b_2$) exceeds domain D
3) n = number of points from Δs_i side
4) Solve equation for r

$$\Delta s_i \left(\frac{1 - r^n}{1 - r}\right) + b_2 = D$$

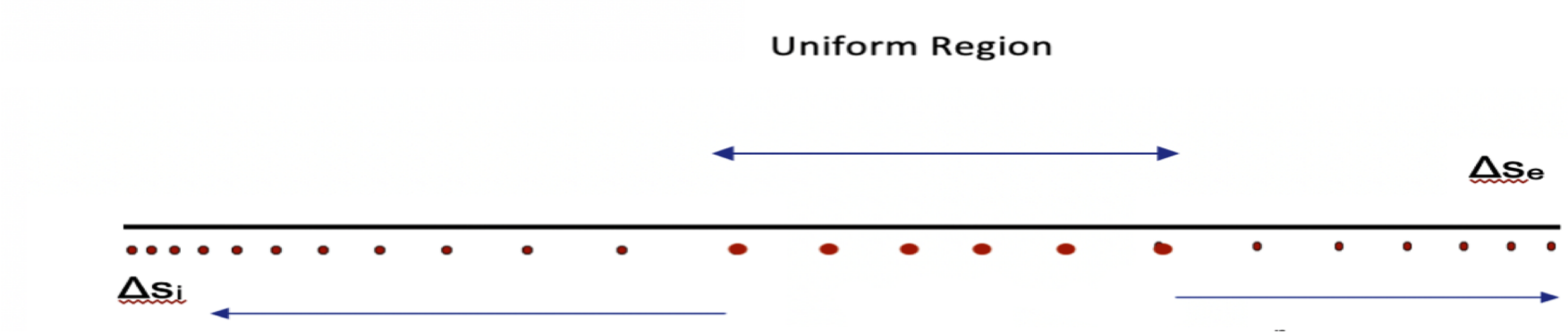


Two-Sided Case 3

1) Build point distribution from Δs_i and Δs_e, stretch using r_{max}, Δs_{max} reached from both sides.
2) Pad the remaining part of domain using Δs_u = min of end spacing from each side
3) n = number of points from Δs_i side
   m = number of points from Δs_e side
   p = number of spacings in uniform region

$$\Delta s_i \left(\frac{1-r^n}{1-r}\right) + p\Delta s_u + b_2 = D$$

Freeze one side solve the equation for r (one-sided case # 2)



Reflections

- A pivotal experience
- Work with the best minds
- Value of teamwork and diligence
- Real-life application
- Appreciation towards mathematical education
- Grateful to mentors, colleagues and everyone in the branch

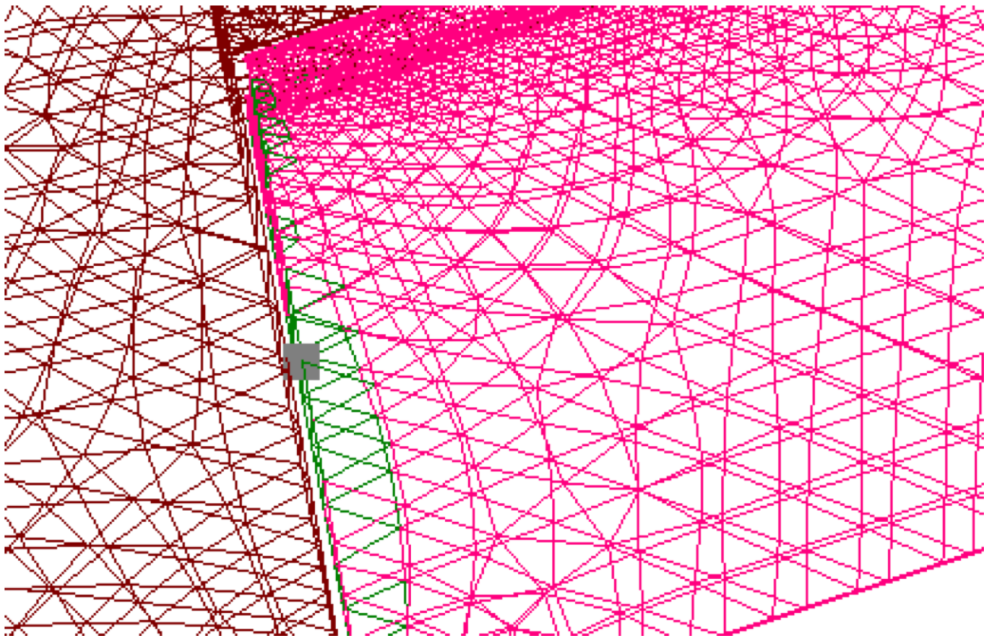
Development of a Triangulation Quality Checker for LAVA

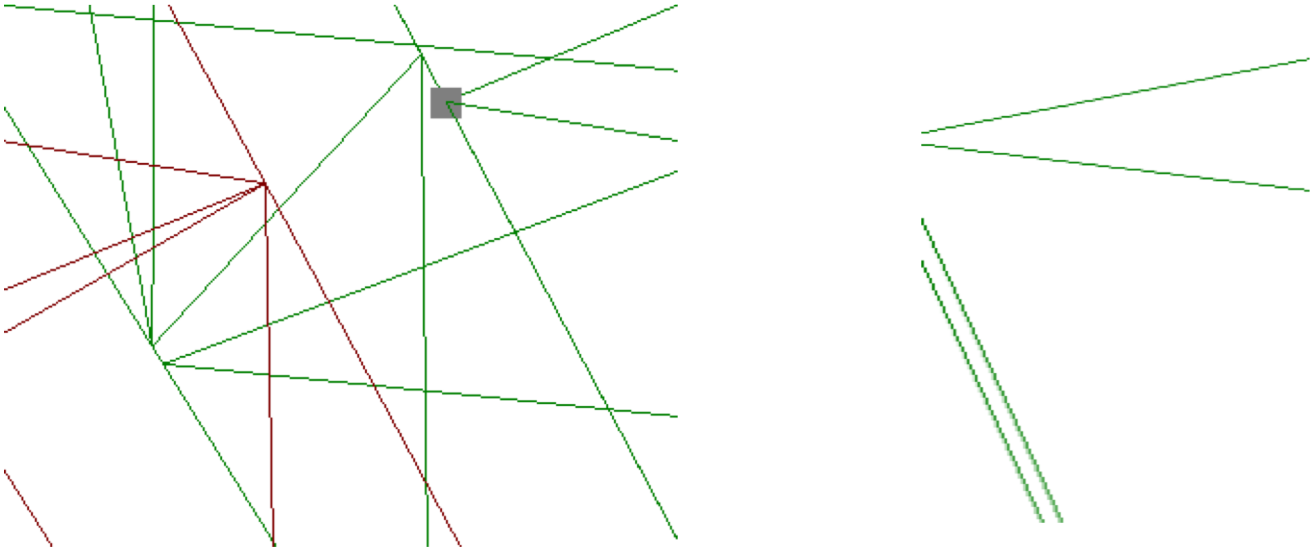
Jacob Zenger
University of Utah

July 30, 2019

Motivation

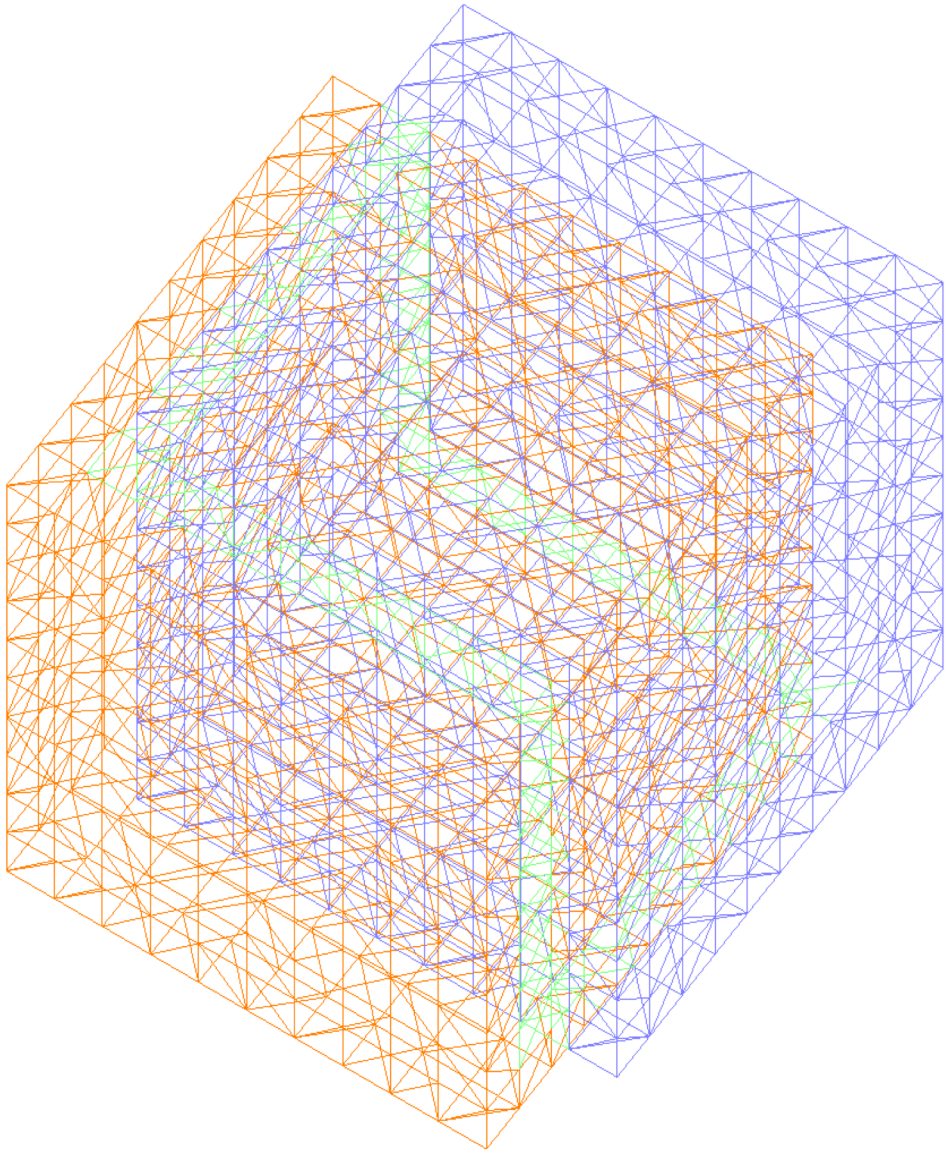
- There is no intersection testing currently available within the LAVA framework
- There are many redundancies when converting from a .stl to a .i.tri
- It can detect things that would be virtually invisible
- It provides a consolidated tool to automatically fix .i.tri files





Code Overview

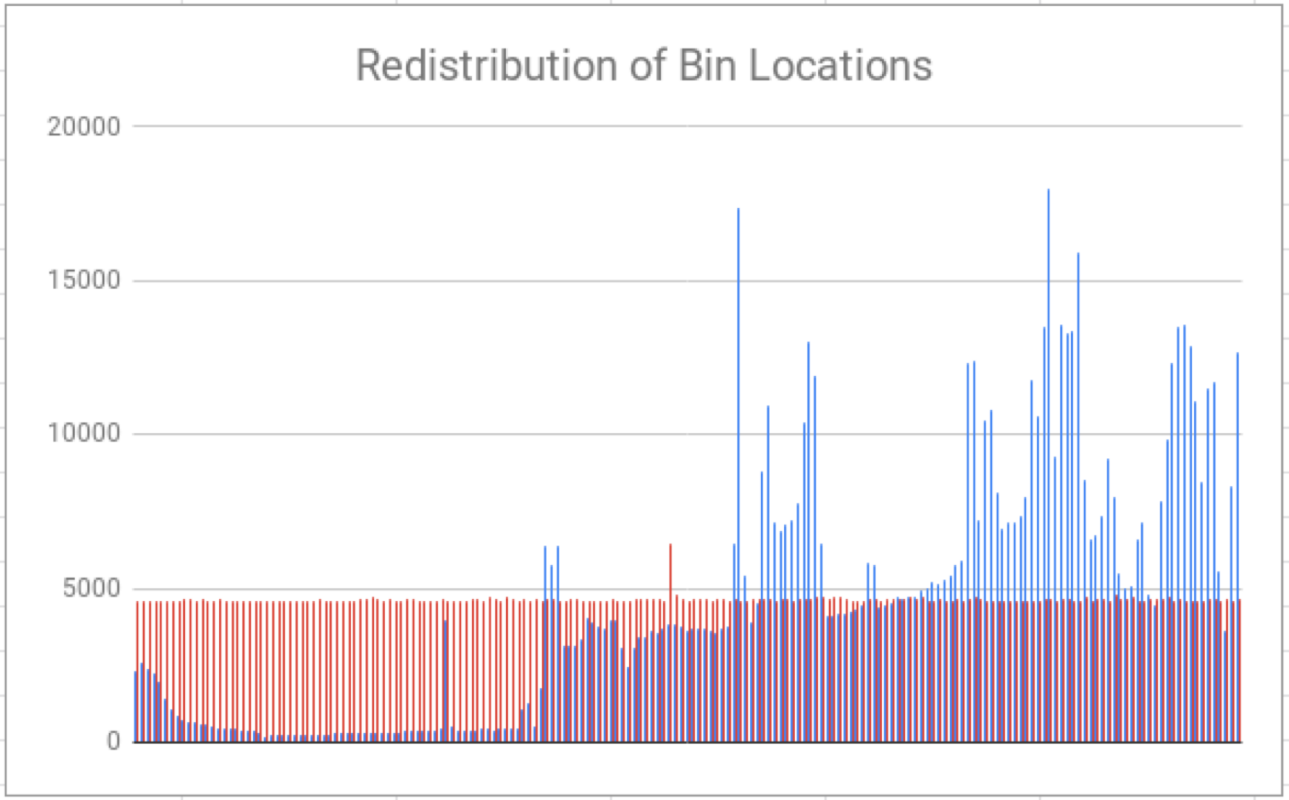
- Input files are either .stl or .i.tri
- Finds any duplicate or unused vertices and eliminates them from the file
- Checks for any intersections that may have occurred in the file
- Fixes any face normal vector errors
- Detects whether an object is open or closed
- Outputs a .i.tri file with any intersections highlighted



Finding Duplicate Vertices

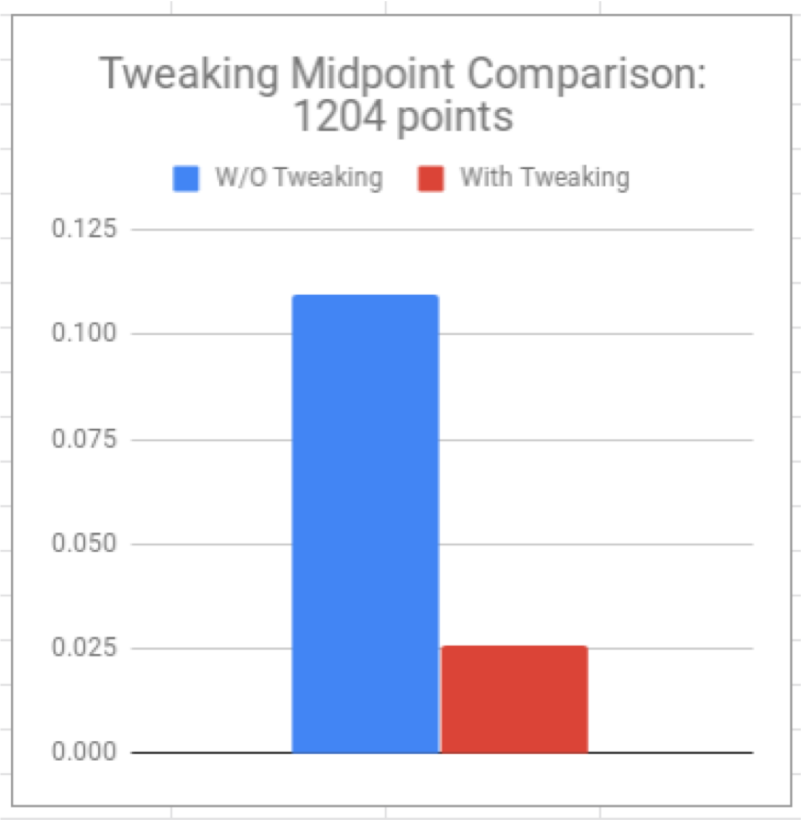
- Many duplicate vertices are created when converting from .stl to .i.tri
- The area gets divided into equally spaced bins
- Bin locations get tweaked to have a more even number of vertices in each one
- Vertices in the same bin are then checked against each other

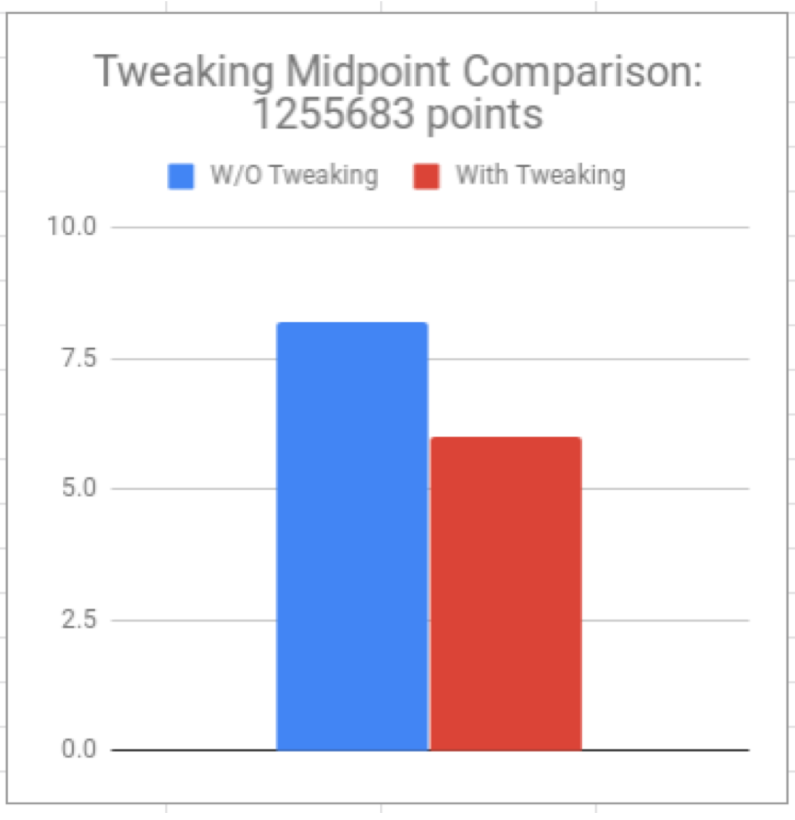
Blue shows the distribution before tweaking the bin locations and red is after tweaking



Timing Results

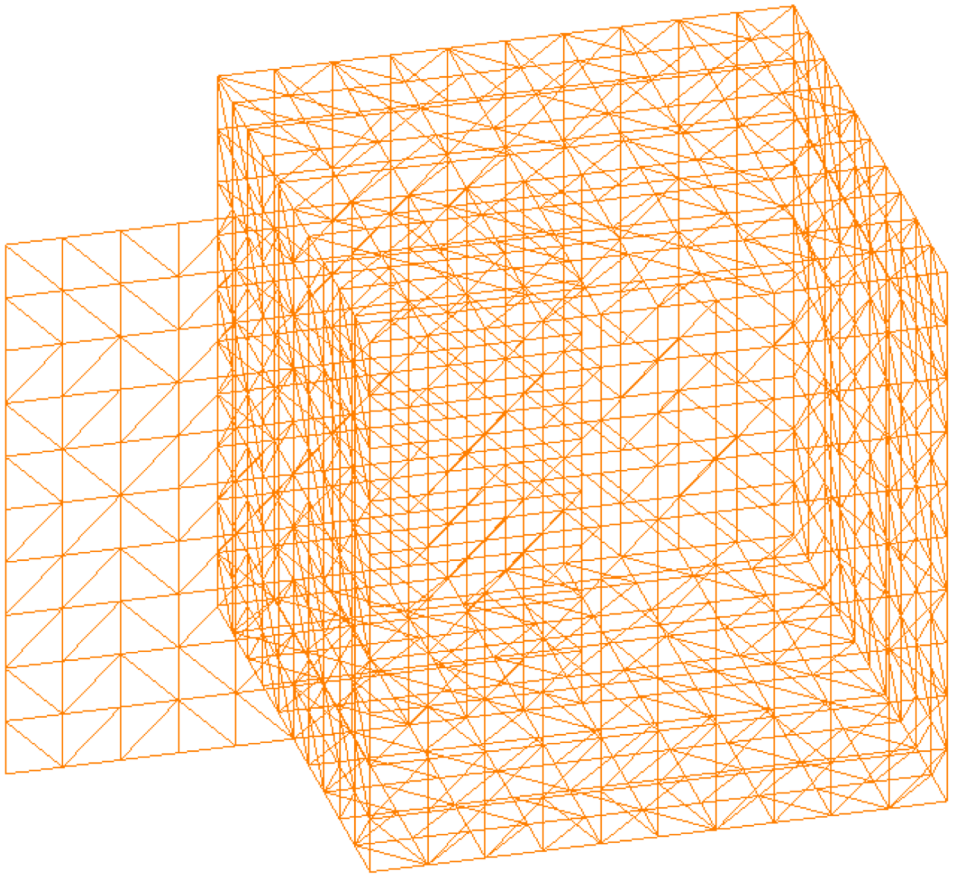
- The largest speed-up in code execution time came from an effective binning of the vertices
- Without binning, a file with 1000 data vertices could take 499500 comparisons to find duplicates but, when binned in 10 bins would be reduced to only 49500 comparisons
- The largest file that was run took over 30 minutes when no tweaking occurred and around 12 seconds when it did





Finding Intersections

- Line/Plane Intersection Testing
  - Convert triangle into a plane
  - Convert line segment into a line
  - Find where the line and plane intersect
  - Check if that point lies within the triangle and the line segment
- Edge Testing
  - Above technique will not catch when an intersection occurs at one of the vertices
  - Checks if an ending edge creates a closed loop or loops – when it does not there is an intersection
  - It checks for any edges that have more than two faces adjacent to



Internship Summary

- I enjoyed using my coding knowledge in a professional setting
- Extended my knowledge in data processing techniques
- Enjoyed working with fellow interns and mentors

Development of a Utility to Automatically Generate Trajectory Files based on User Input

Keshav Sriram
Diamond Bar High School, Grade 12

July 30, 2019

Purpose

- This trajectory file generator is a Python script that takes two XML files which specify the components and their motions as input and outputs a trajectory file which can then be used to model that motion in a flow solver such as LAVA

- The XML file is based on the one created by William Chan in *An Interface for Specifying Rigid-Body Motions for CFD Applications*

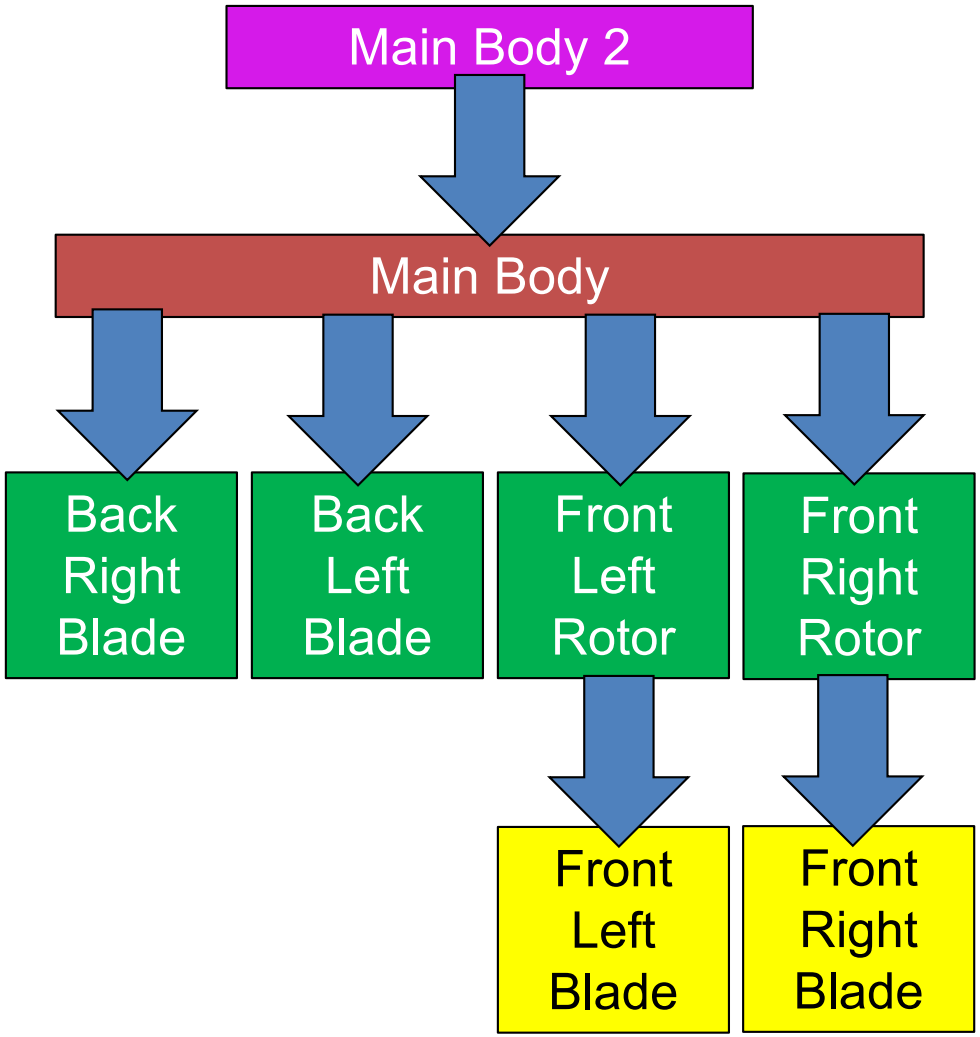
- The interface will eventually be used to create trajectory files for rotorcraft

Outline

- Prescribed Motion
  - Config.xml
  - Scenario.xml
- Trajectory File
- Examples
  - Drone
  - Bee
- Internship Summary

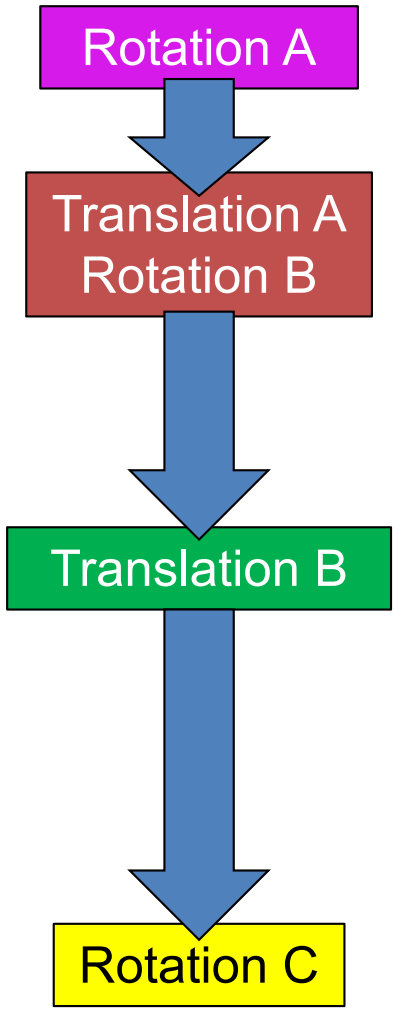
XML Files

- Config.xml
  - Lists all structures and containers used in the file
    - Structure type objects are represented by actual grids
    - Containers represent a group of grids or an additional motion applied to a grid
    - Trajectory files are created for structures but not for containers
  - List parent of each item to create a hierarchy
    - Hierarchies are represented using hierarchy charts
    - All motions prescribed for objects higher up in the hierarchy apply to all of their children as well



XML Files

- Scenario.xml
  - Lists all prescribed motions for the objects in the Config.xml file
  - Two types of prescribed motions: rotations and translations
    - Rotations must be specified with a rotation axis, center, and a speed of rotation
    - Translations must be specified using the x, y, and z components of the velocity of translation.
  - Motions are applied in a certain order
    - Motions of higher order components are applied before motions of lower order components
    - Rotations are applied before translations



Rotation A

Rotation A
Rotation B
Translation A

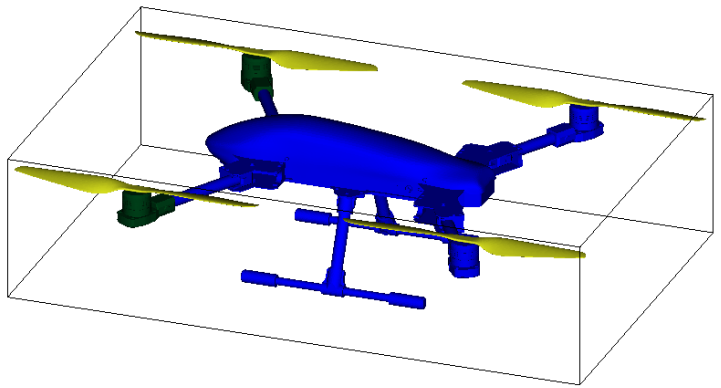
Rotation A
Rotation B
Translation A
Translation B

Rotation A
Rotation B
Translation A
Translation B
Rotation C

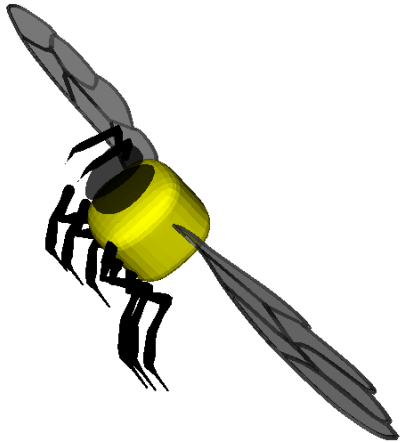
Trajectory File

- 7 columns of data
- Time
  - Shows the time from the start of the motion
  - Organized in a constant interval (Δt)
- Translation
  - Three columns: Δx, Δy, Δz
  - Displacement of the object center from its position at t = 0
- Rotation
  - Three columns: θx, θy, θz
  - Angles are in radians
  - Rotation of the body axes about the x, y, and z axes
  - Order of rotation:
    - About the z-axis
    - About the y-axis
    - About the x-axis

Drone Example



Bee Example



Internship Summary

- It was a great learning experience working on difficult problems such as this one
- This internship greatly increased my expertise in coding and taught me several more programming languages
- This internship helped me get much more interested in programming than I had been before
- It was fun to meet new people with similar interests to mine and talk to them about their fields of expertise
- Thanks to James Jensen, William Chan, Shishir Pandya, Gerrit Stich, Cetin Kiris, and all the other Computational Aerosciences interns for everything you have done.

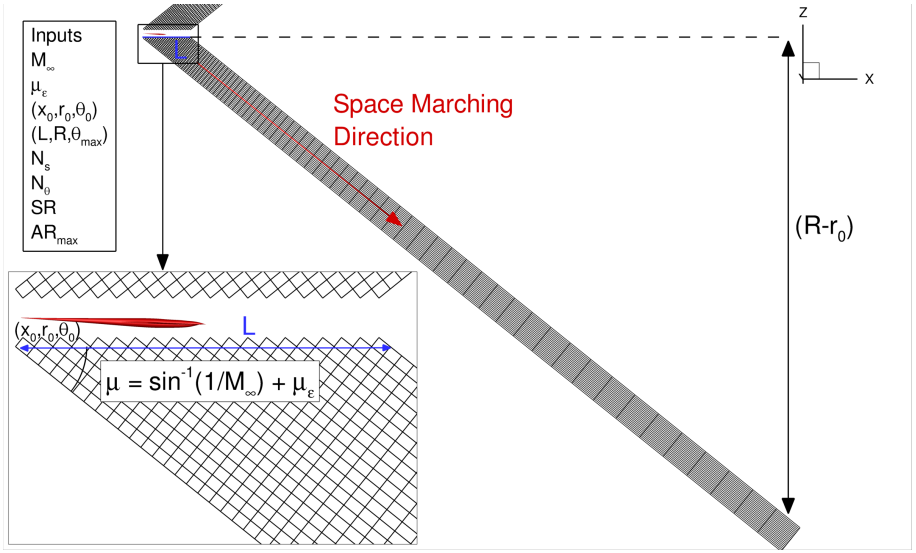
Development of a Mesh Redistribution Algorithm for Structured Curvilinear Meshes

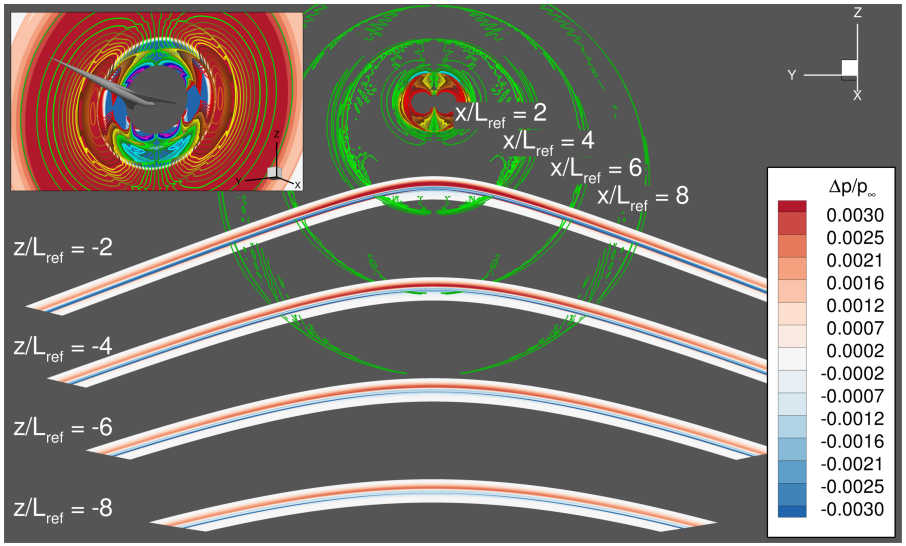
Chase Ashby
University of Kentucky

07/30/2019

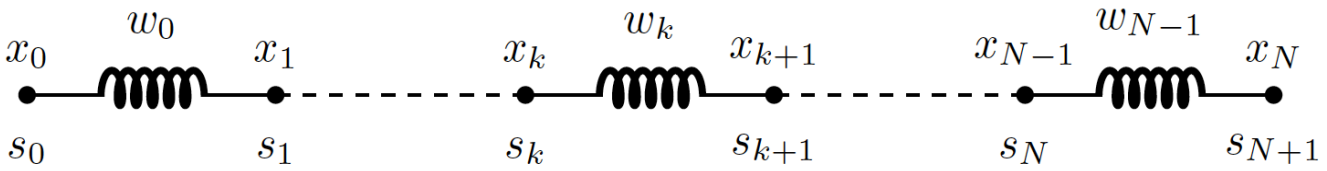
Motivation

- Tasked with developing mesh redistribution algorithm for implementation in LAVA
- Compared 1D equidistribution algorithm to nonlinear programming (NLP) approach
- Redistribution maintains computational costs while increasing accuracy





Equidistribution Approach



Input: CFD solution, mesh
Output: Redistributed mesh
initialization;
while $i < numAdaptations$ **do**
  $currMesh \leftarrow$ current mesh;
  $w_k \leftarrow$ spring constants ;
  $w_k = \text{smooth}(w_k)$;
  $A \leftarrow$ tridiagonal matrix ;
  mesh $\leftarrow solveTri(A)$;
  Interpolate data to new mesh;
  Compute approximations;
  if $\|currMesh - mesh\|_1 < tol$ **then**
    return mesh;
  else
    adjust β;
  end
end

$$w_k \Delta s_k = C$$

Balance equation:

$$w_k \left(s_{k+1} - s_k\right) - w_{k-1} \left(s_k - s_{k-1}\right) = 0$$

Definition of weights:

$$w_k = F_k + \beta \frac{|f_{\text{exact},k} - f_{\text{approx},k}|}{e_{\max}}$$

Nonlinear Programming Approach

$$minimize: \sum_{k=1}^{N-1} \left(f_{exact,k} - f_{approx,k}\right)^2 \left(x_k - x_{k-1}\right)^2$$

$$subject\ to: \ x_{k-1} \leq x_k$$

- Implemented using **pyOptsparse**
- Optimizer: **SLSQP**
  - Han-Powell quasi-Newton method
  - Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS)

Testing

$f(x) = e^{\sin(x)}$

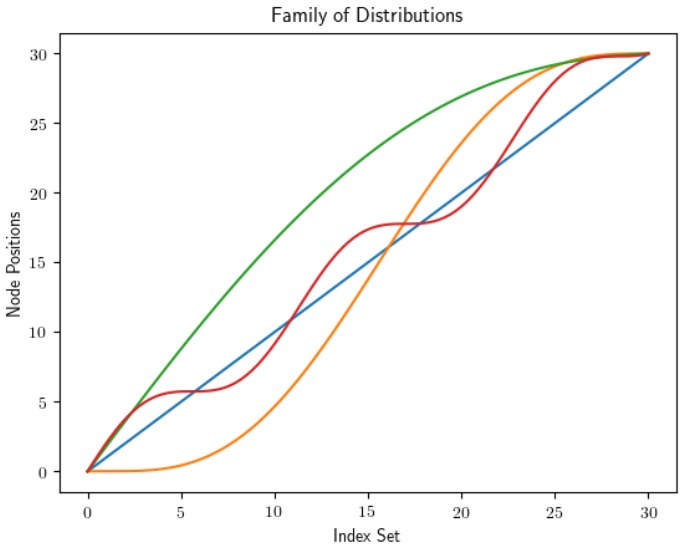
Equidistribution	Nodes	L2 Error	Computational Time (s)
	31	1.52E-02	0.005
	100	3.33E-05	0.01
	500	8.72E-09	0.179
	1000	2.63E-10	1.0
NLP	Nodes	L2 Error	Computational Time (s)
	31	1.74E-02	0.84191
	100	1.91E-02	23.77725
	500	1.88E-02	1223.12
	1000	1.89E-02	8037.695

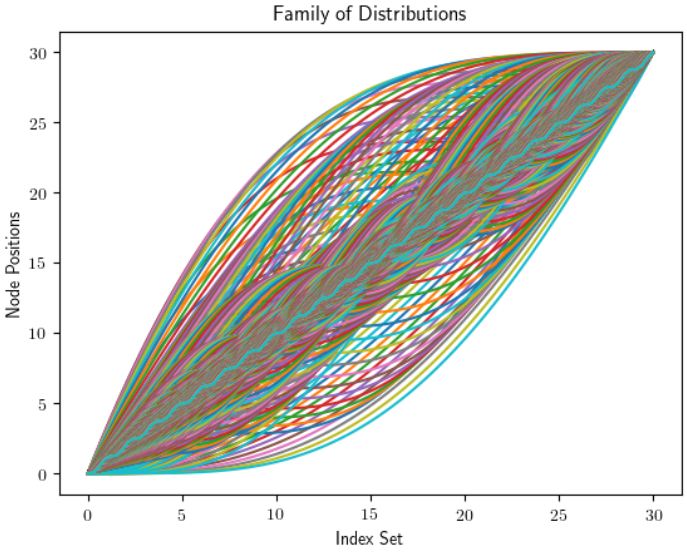
Uniform mesh with 1000 nodes: **5.15E-10**

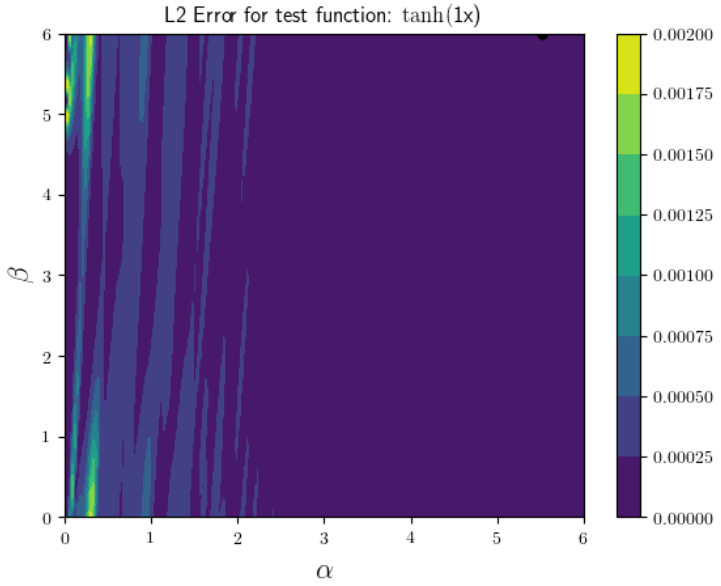
$$f(x) = 1 - \frac{C_1}{4C_2}\left(\tanh(x-5)+1\right)\left(\tanh(x-20)-1\right)\cos\left(\frac{\pi}{5}(x-10)\right),\ C_1 = 10630.2,\ C_2 = 1000.0$$

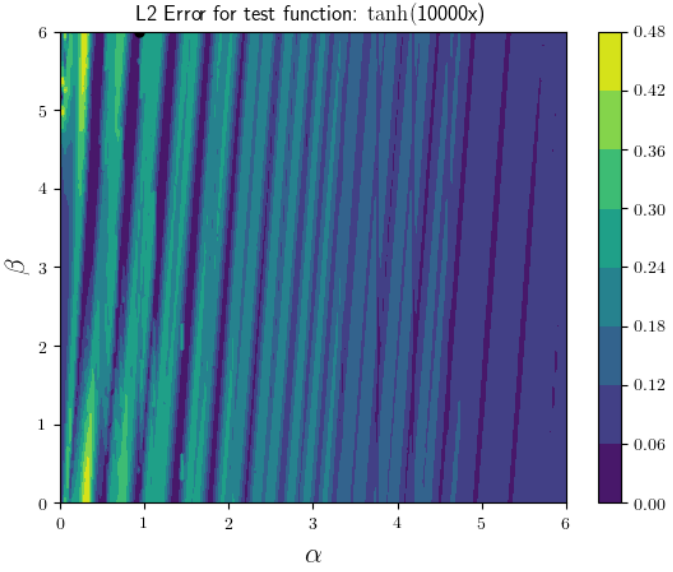
Equidistribution	Nodes	L2 Error	Computational Time (s)
	31	2.85E-02	0.01
	100	5.36E-09	0.043
	500	1.67E-12	0.334
	1000	5.20E-14	1.201
NLP	Nodes	L2 Error	Computational Time (s)
	31	1.08868E-05	0.98335
	100	1.11841E-05	4.1597
	500	1.15804E-05	225.3964
	1000	1.11817E-05	1732.433

Testing

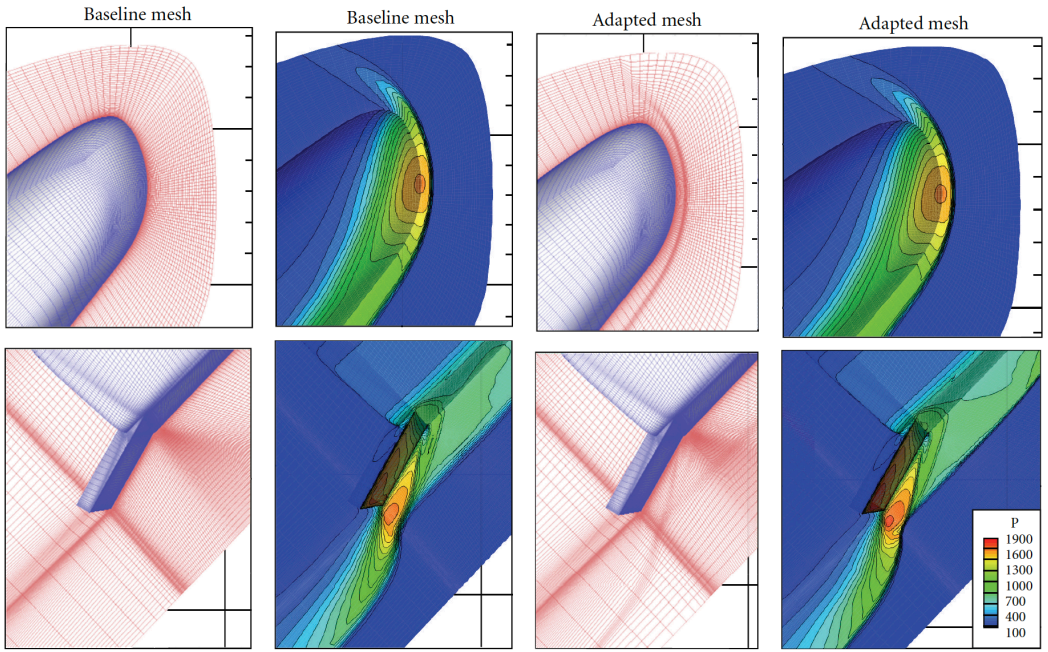

Family of Distributions
Node Positions
Index Set
0
5
10
15
20
25
30

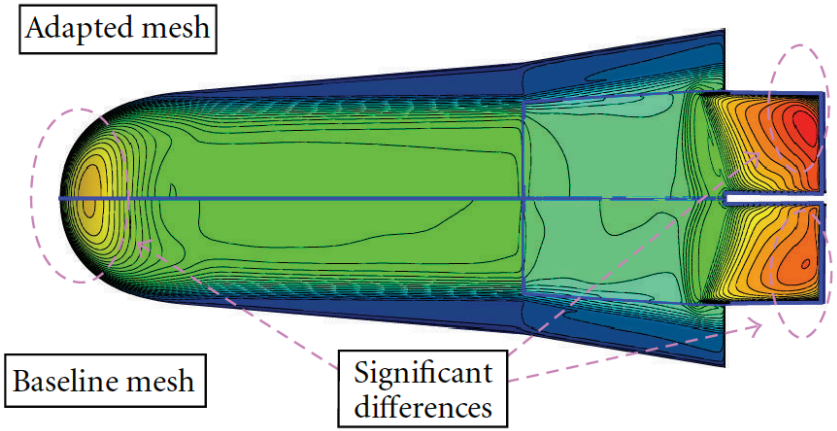

Family of Distributions
Node Positions
Index Set
0
5
10
15
20
25
30


L2 Error for test function: tanh(1x)
α
β
0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000


L2 Error for test function: tanh(10000x)
α
β
0.48
0.42
0.36
0.30
0.24
0.18
0.12
0.06
0.00

Conclusions/Future Work





"*The Role of Mesh Generation, Adaptation, and Refinement on the Computation of Flows Featuring Strong Shock*," Bonfiglioli, A., Paciorri, R., and Di Mascio, A., **Modeling and Simulation in Engineering**, doi:10.1155/2012/631276.

Internship Experience

- **Reflections:**
  - Gained practical experience in Python and Fortran
  - Explored Calculus of Variations, tensor analysis, Riemannian Geometry
  - Great team and working environment

Acknowledgements: Dr. Jeff Housman, Dr. Gaetan Kenway, Gerrit-Daniel Stich, James Jensen, Dr. Cetin Kiris, Keshav Sriram, Jacob Zenger

THANK YOU!

Uncertainty Quantification with Cart3D and QUEST

Liam Smith (Georgia Tech)

Mentor: Marian Nemec

30 July 2019

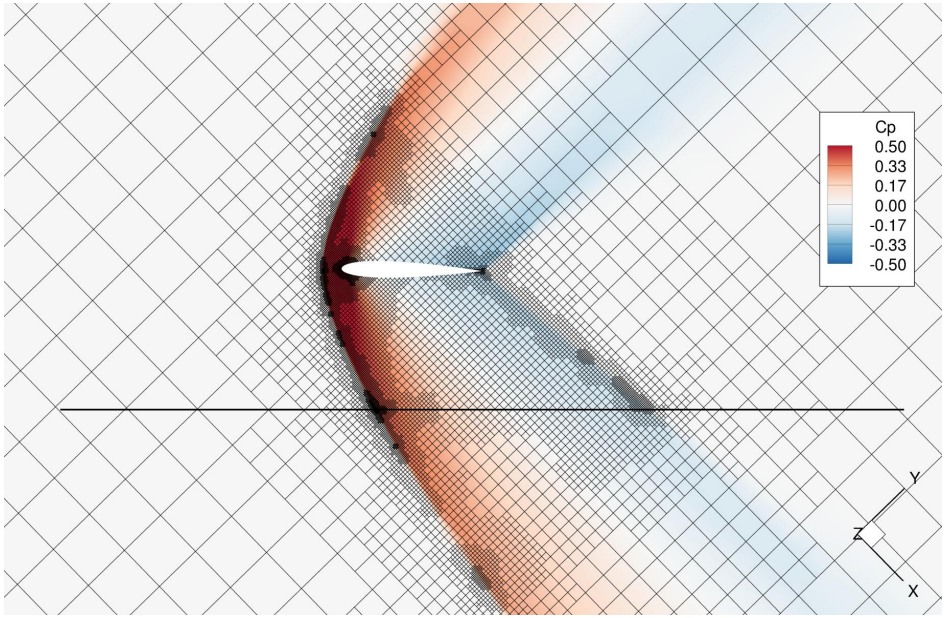
Background and Motivation

- Cart3D
  - CFD analysis package utilizing adjoint-driven mesh adaptation
  - Given specific geometry and flow conditions we can efficiently compute aerodynamic performance of an aircraft
- Actual operating conditions are never known exactly
  - Can we quantify the uncertainty in aircraft performance given uncertainty in operating conditions?
- QUEST - Quantified Uncertainty with Error bounds Software Toolkit
  - Developed by Timothy Barth
  - Quantifies the uncertainty of engineering outputs, such as lift and drag, pressure signatures and surface pressure distributions
  - Couples CFD discretization error with statistical error estimates
- Objectives:
  - Integrate Cart3D with QUEST
  - Characterize the uncertainty in near-body pressure signatures of supersonic aircraft

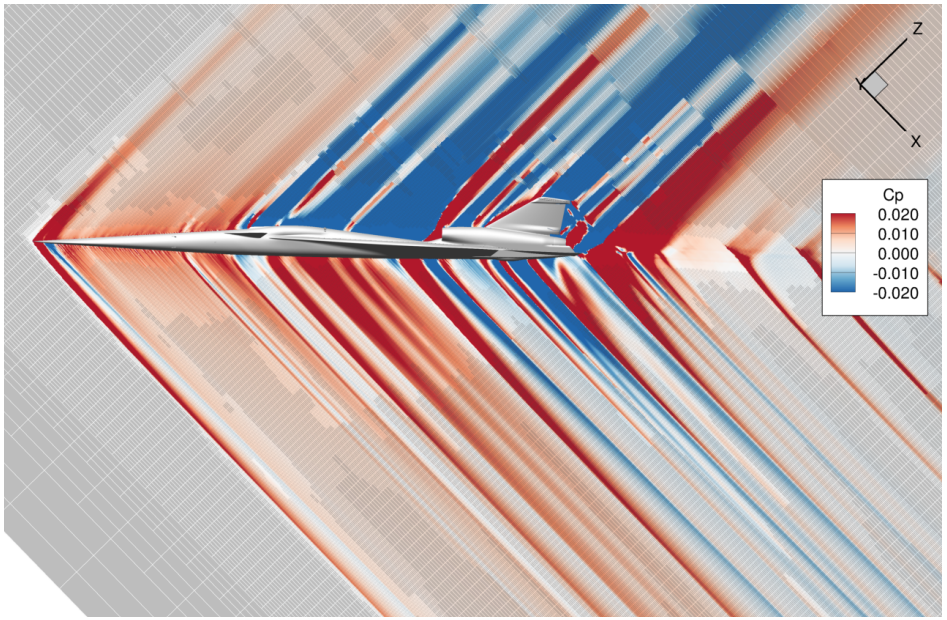
Selected Examples

1) NACA 0012 Airfoil
   - Nominal Flow Conditions
     - Mach = 1.4
     - Alpha = 1.0°

2) Low-Boom Flight Demonstrator (LBFD)
   - Nominal Flow Conditions
     - Mach = 1.4
     - Alpha = 2.1°



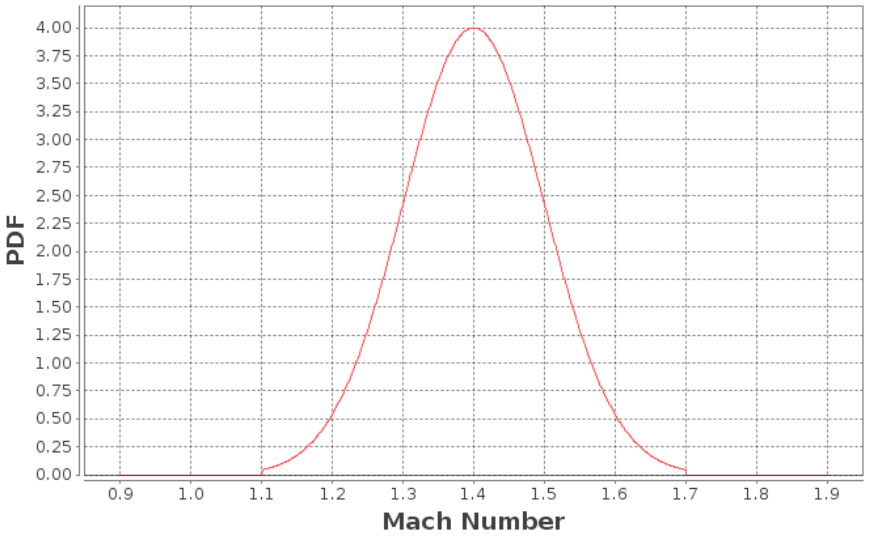
- Output: Pressure Distribution
  - NACA 0012: Line Sensor 1 body length below airfoil
  - LBFD: Line Sensor 3 body lengths below aircraft

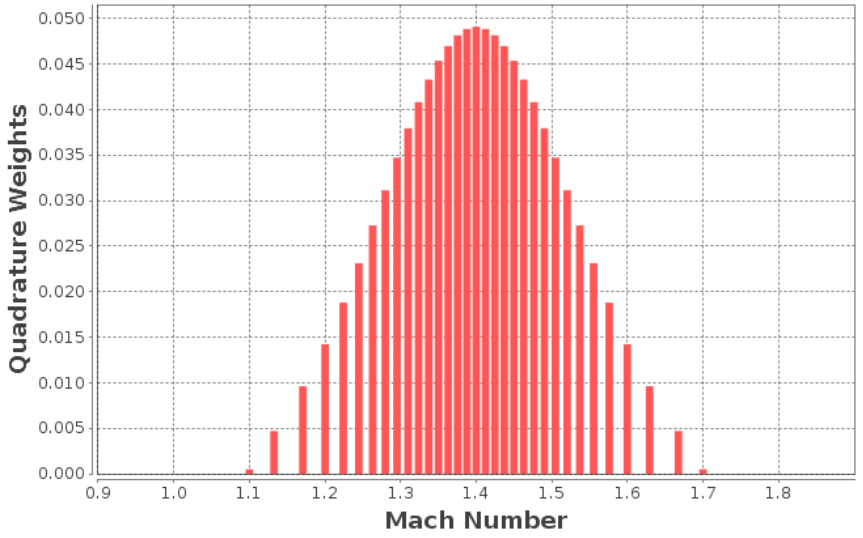


QUEST Overview

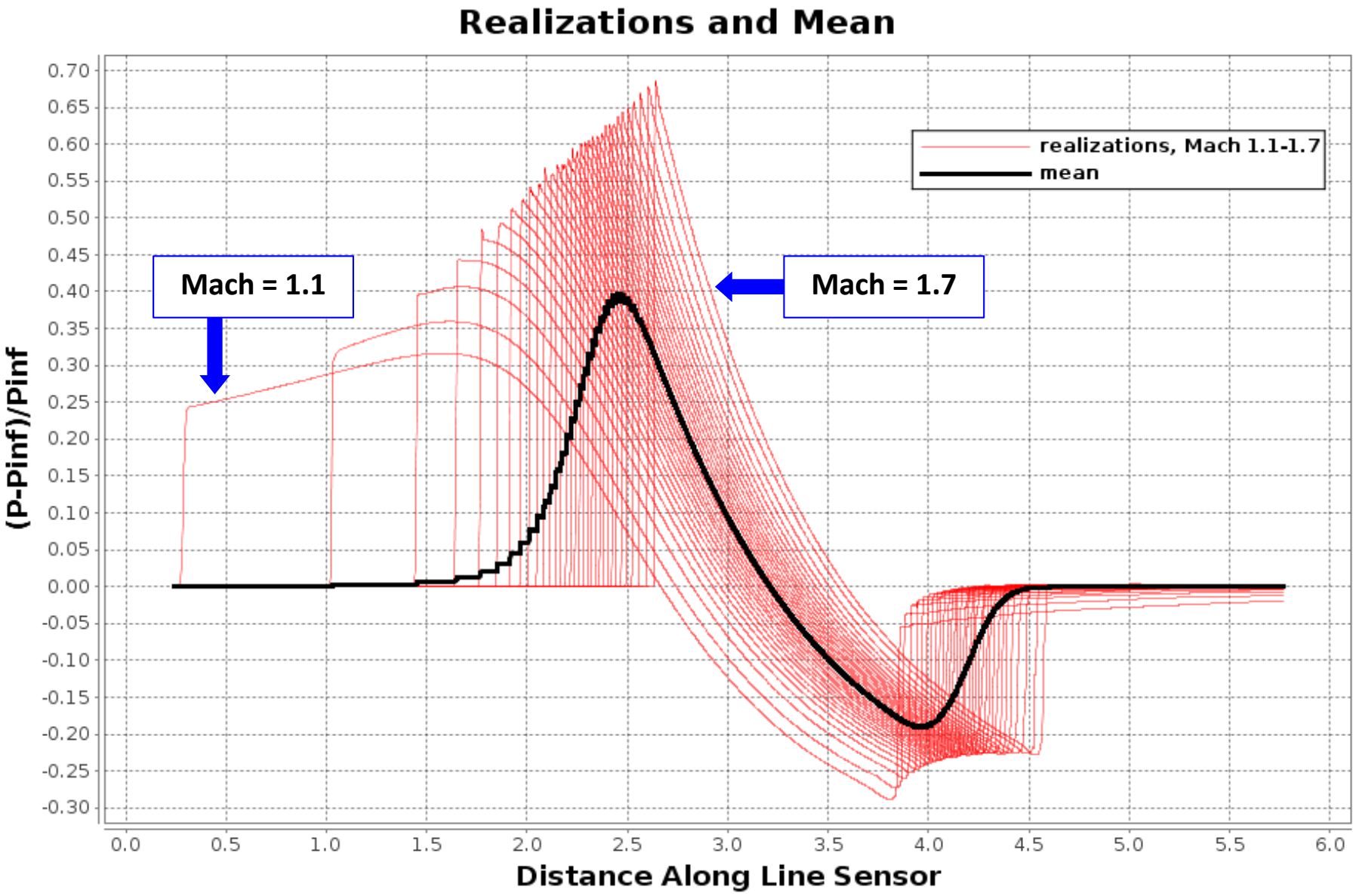
- QUEST Prep
  - Specify uncertainty parameters and distributions
    - Mach number
      - Normal probability density function (PDF)
      - NACA 0012: Mean = 1.4, σ = 0.1
  - Choose a method
    - Nested Dense Clenshaw-Curtis Quadrature (6 levels)
  - Database of individual CFD runs (realizations)
    - NACA 0012: 33 realizations
- QUEST Post
  - Input CFD results for each realization
  - Compute uncertainty statistics

NACA 0012

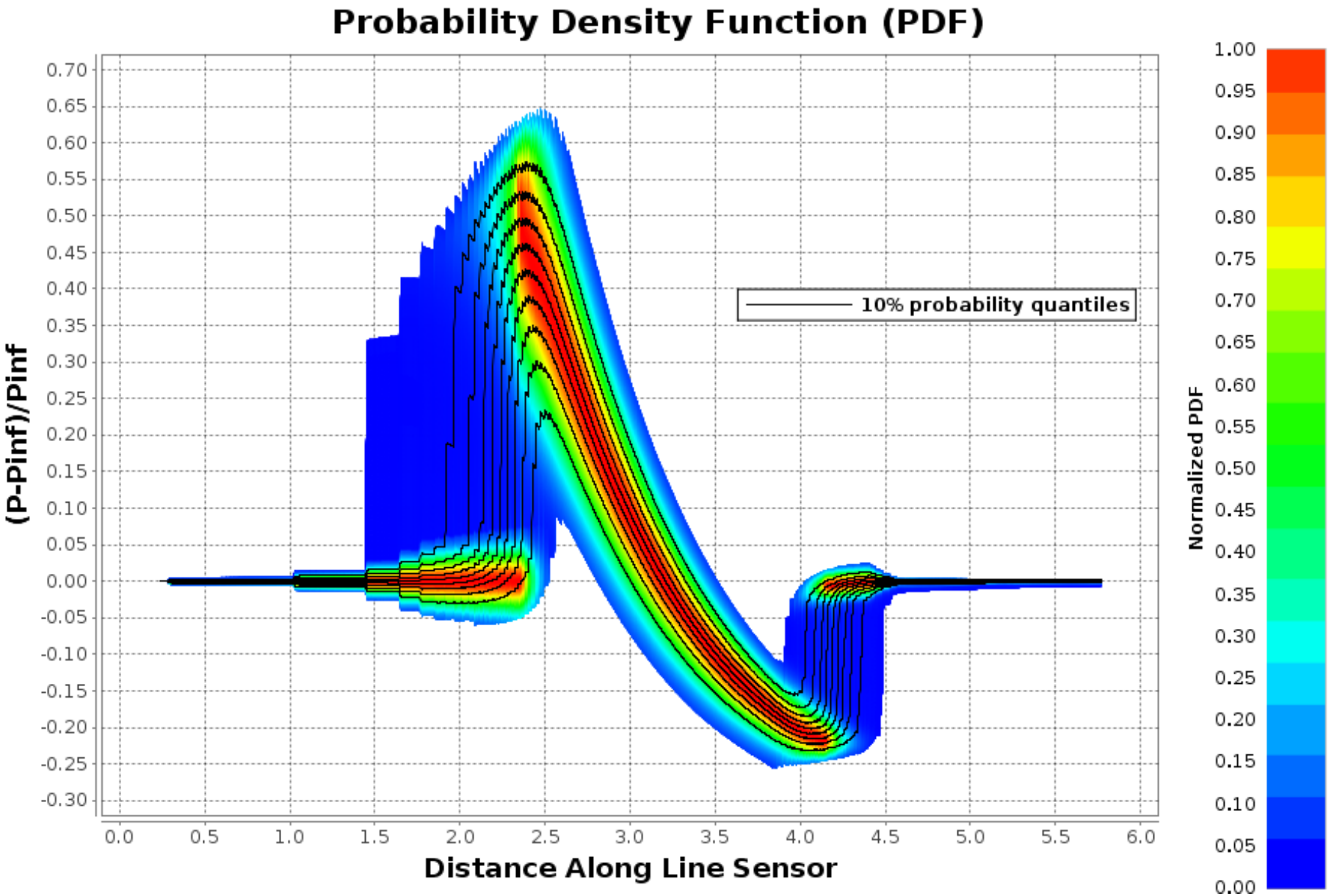




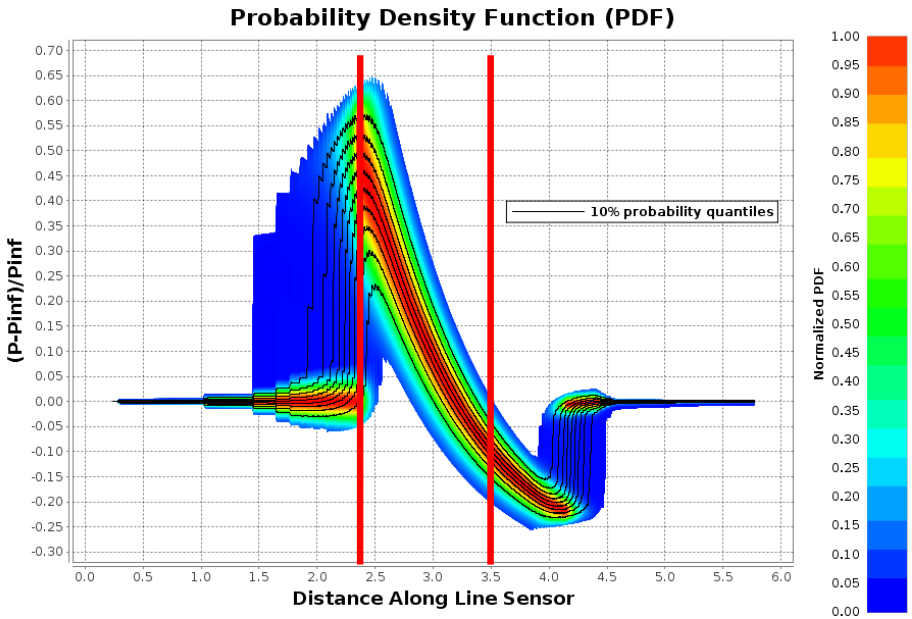
NACA 0012 - Realizations

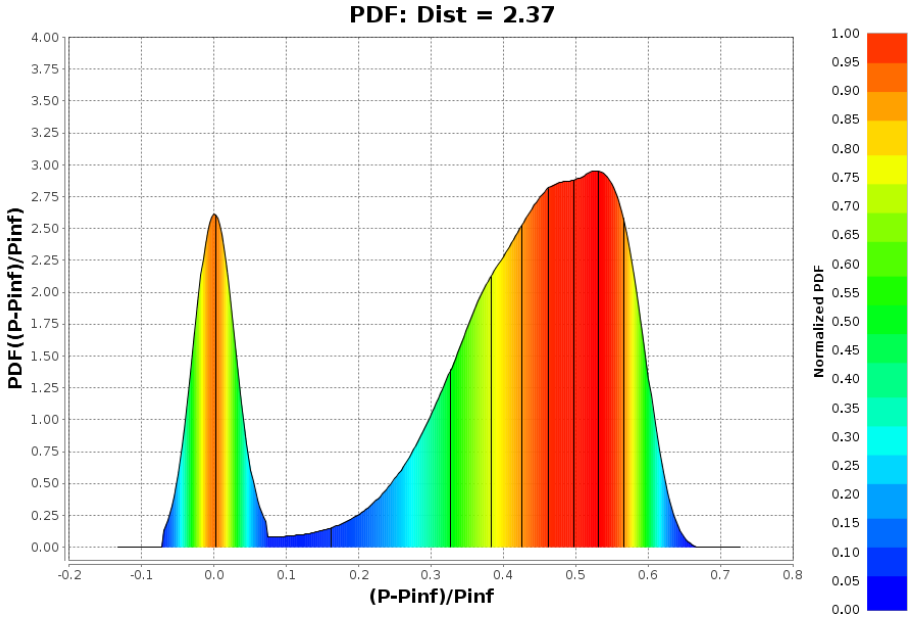


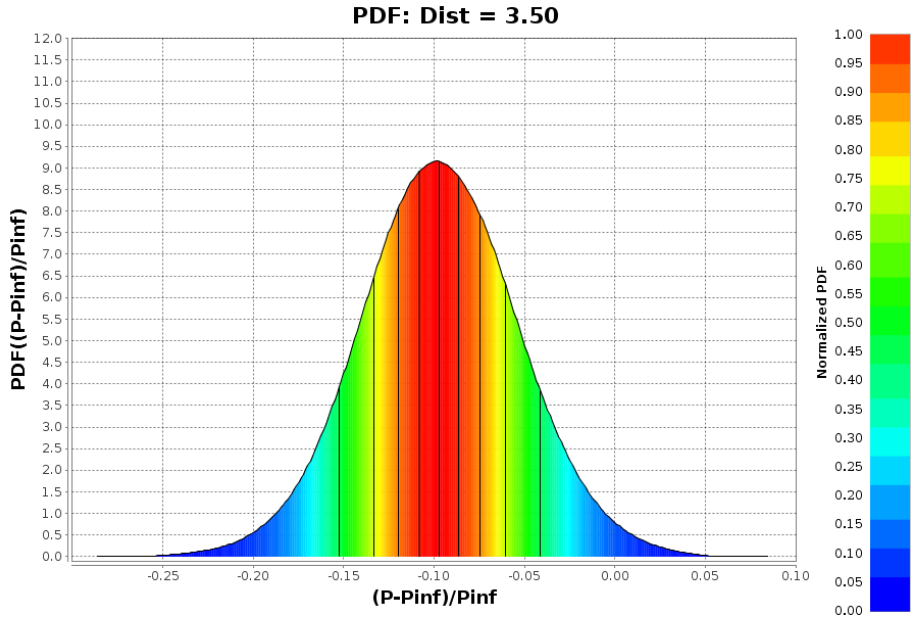
NACA 0012 – Aggregate PDF



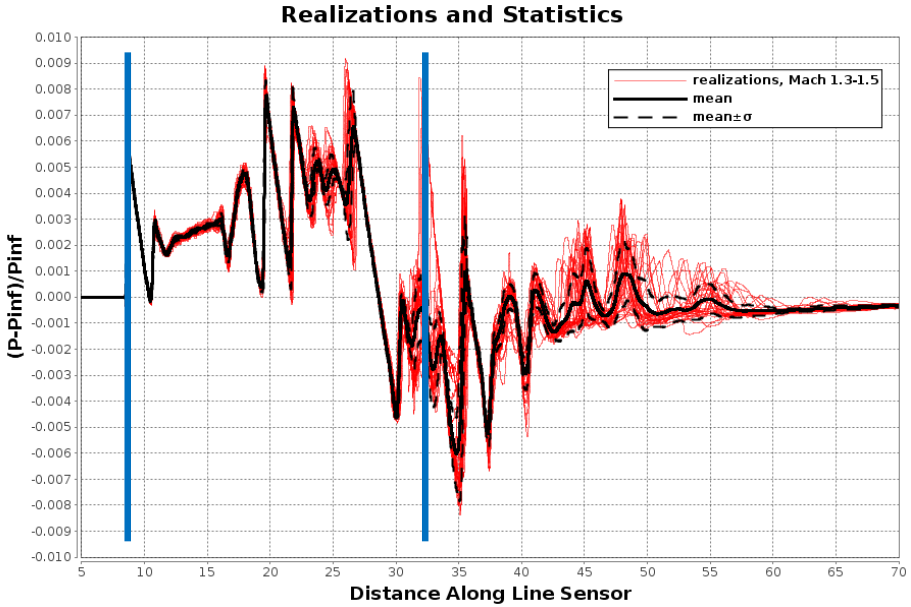
NACA 0012 – PDF Slices

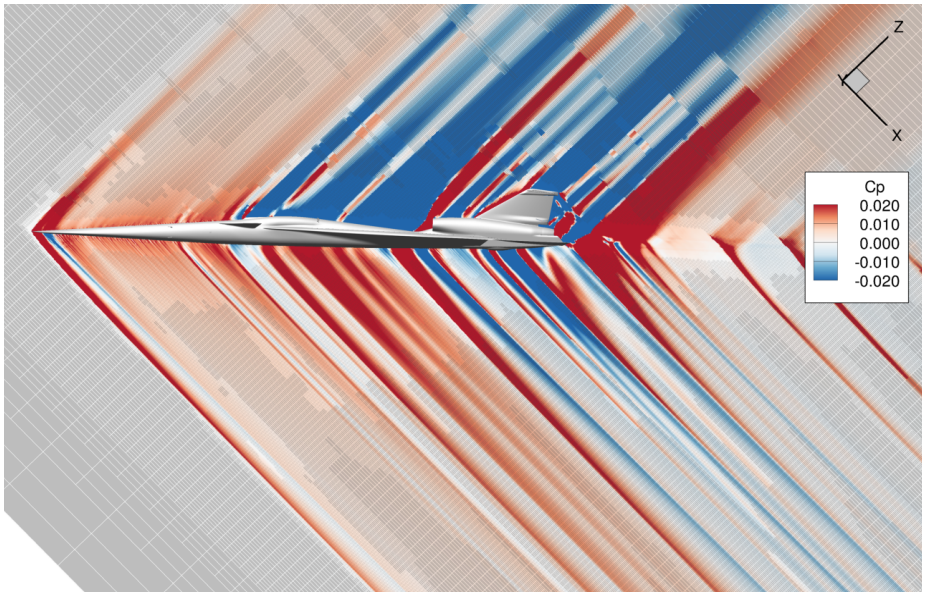


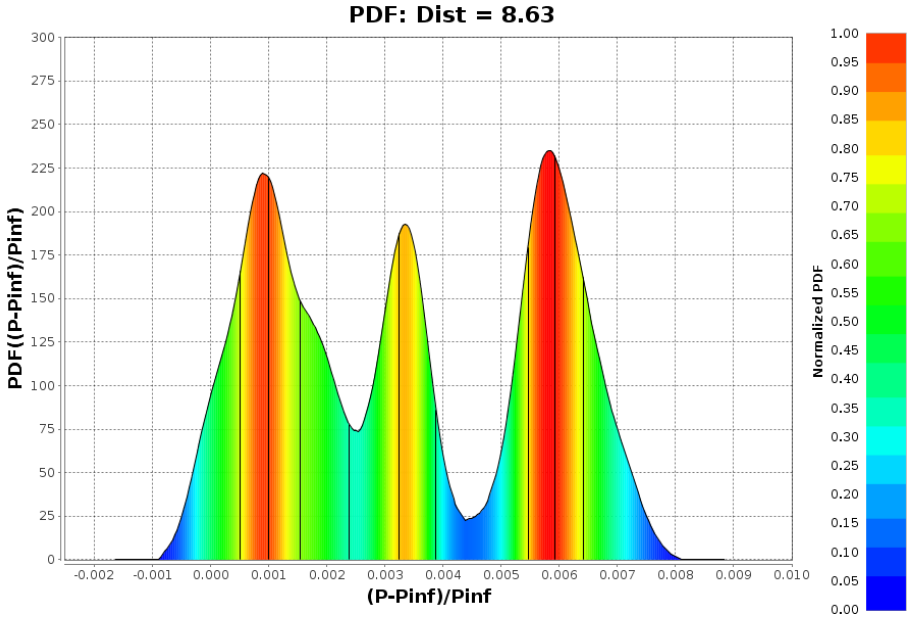


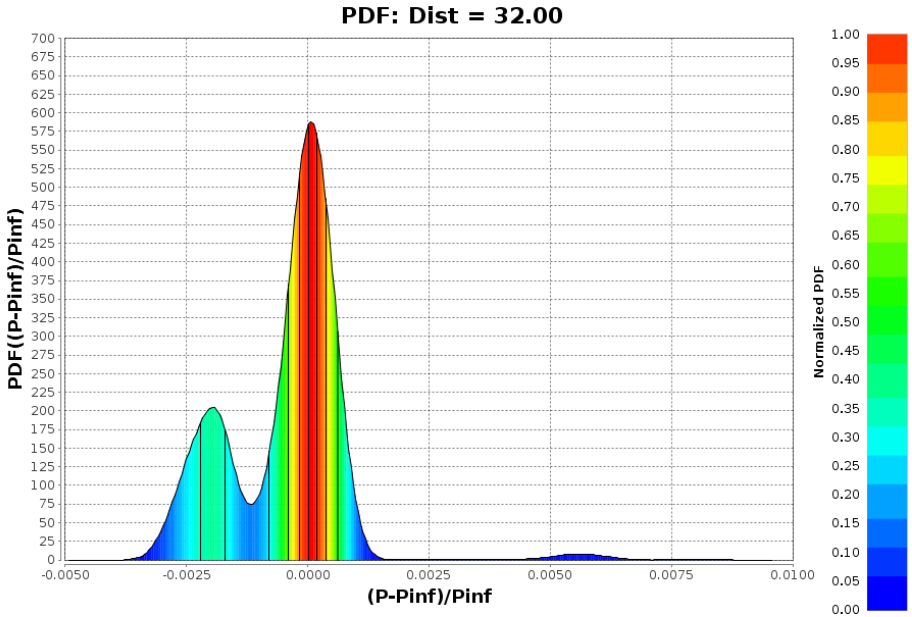


LBFD – PDF Slices









Conclusion and Future Work

- Demonstrated ability to characterize uncertainty in pressure distributions along line sensors
  - PDF plots allow visualization of most likely pressure values given uncertainty in flight conditions
- Applicable to complex problems such as the LBFD
  - Could be very useful in characterizing uncertainty in the ground-level noise profile due to variations in flight and atmospheric conditions
- Future Work
  - More sources of uncertainty
    - Control surface deflections
    - Aeroelastic deformations
    - Atmospheric properties
  - Apply to ground signatures and level of noise

Questions?

Unstructured CFD Support for Commercial Supersonic Technology Wind Tunnel Tests

Robert Comstock
Cal Poly San Luis Obispo, CA

and

William Bowes
University of California, Davis

July 30th, 2019

Abstract and Bio

William Bowes

- Senior at University of California, Davis
- Mechanical engineer undergraduate
- Experience in multiple industries such as Velodyne and Tesla
- Structures lead for UCD Formula Racing engineering team

Robert Comstock

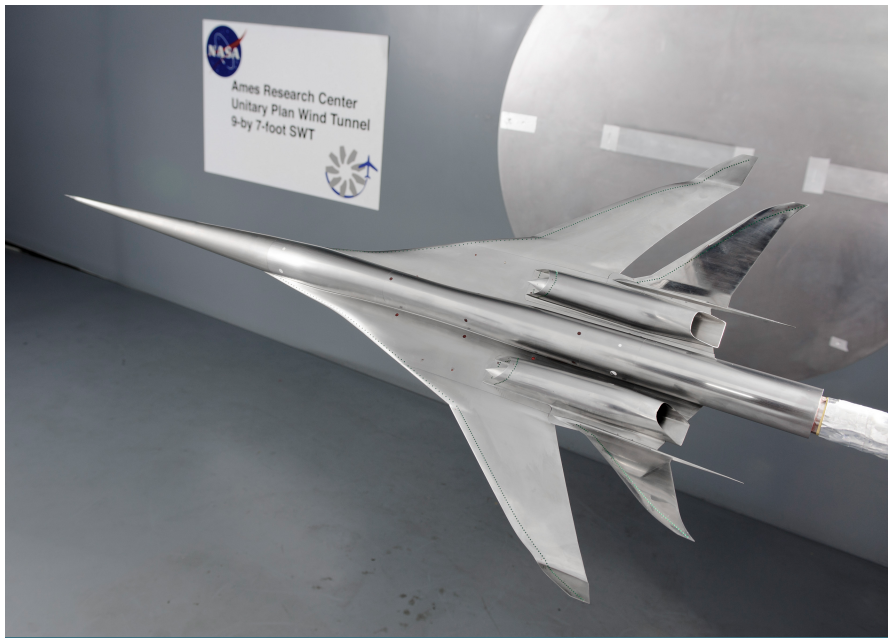
- Graduate student at California Polytechnic State University San Luis Obispo
- Concentration in aeronautics
- Worked remotely for the LAVA group during the school year and on-site during the summer
- Aero lead for Cal Poly Formula SAE

Abstract:

The NASA Ames wind tunnel division has been testing Commercial Supersonic Technology (CST) design concepts in different wind tunnel test sections. William and Robert from the LAVA group provided CFD support by running simulations of upcoming wind tunnel tests with unstructured grids generated from STAR-CCM+ and Pointwise. Near-body pressure signatures were measured from the simulations to predict the impact that newly designed struts in the test sections may have on the test data for the CST concept models.

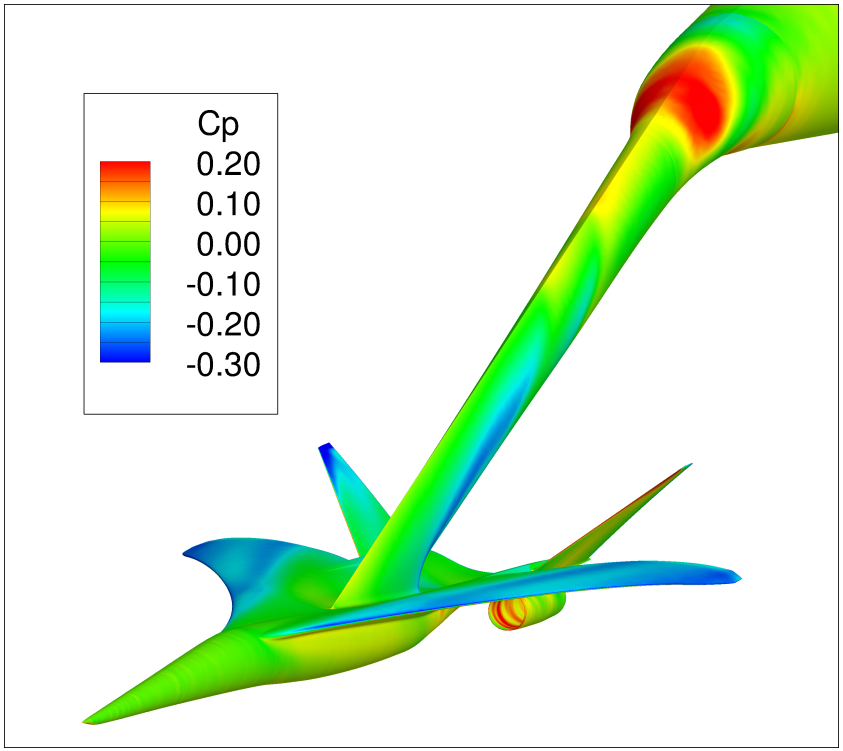
Commercial Supersonic Technology (CST)

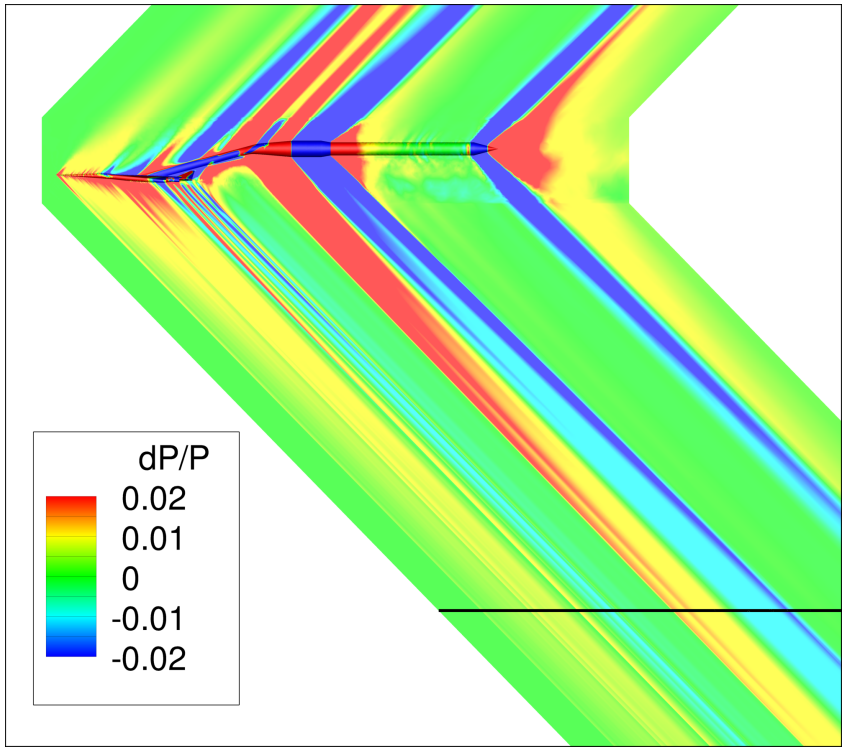
- Aimed at developing technologies in order to enable commercial supersonic flights over land
- Entails the development of low boom supersonic aircraft designs as well as technologies that make these advanced designs possible
- Project is multi-disciplinary covering all aspects of aircraft design



LAVA Team's Contribution to CST

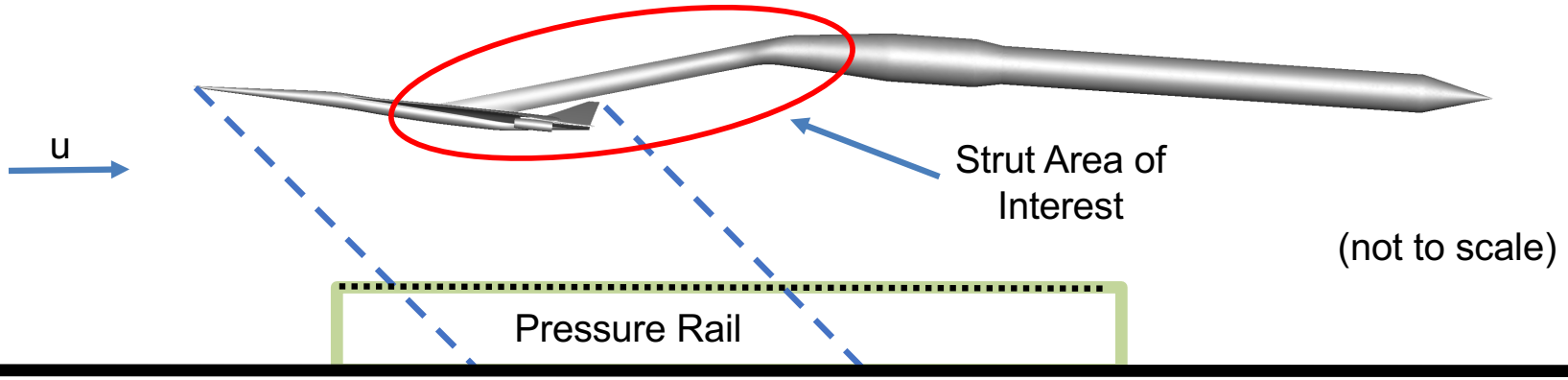
- The LAVA team contributes to the CST project by developing CFD methodologies for simulating low boom supersonic aircraft
  - Help support wind tunnel tests by performing CFD prior to the experiment
  - Utilize the wind tunnel data from the experiments to validate and further develop computational tools specifically aimed at aerodynamic and sonic boom performance





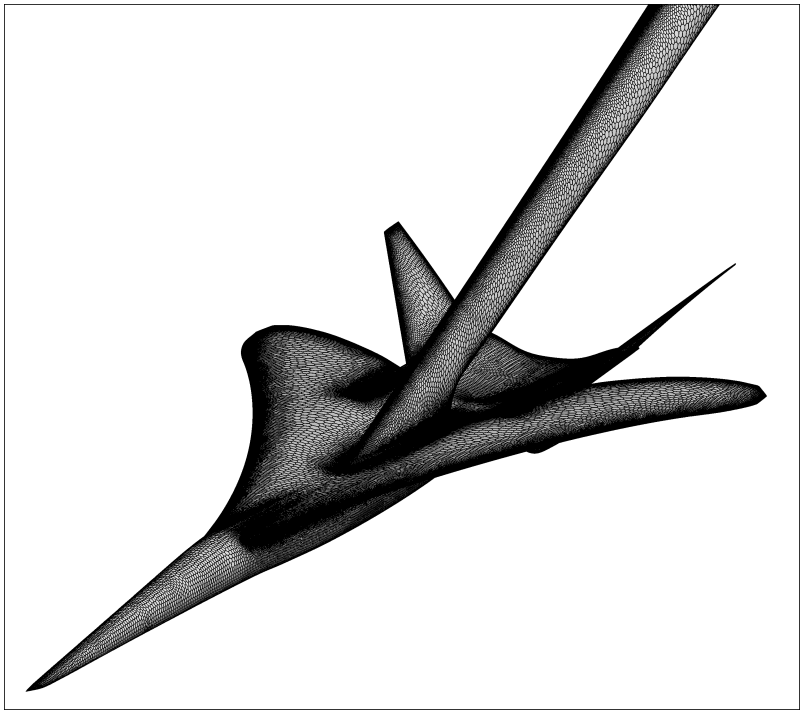
Upcoming Wind Tunnel Test

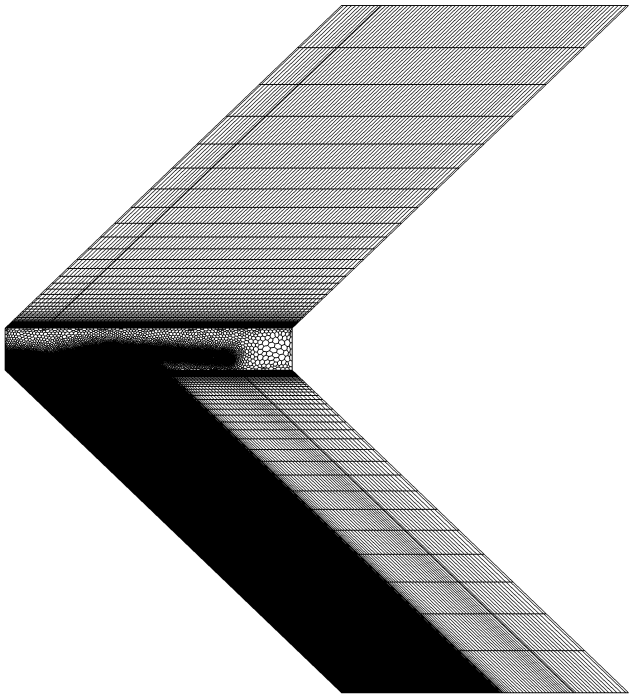
- Upcoming wind tunnel test of Low Boom Flight Demonstrator (LBFD)
  - 11ft x 11ft test section at ARC Unitary Wind Tunnel
- Strut needs to be redesigned to withstand side force loads
  - A previous test in the 11ft x 11ft test section with a CST model showed that the strut had large oscillations
  - Strut must also have minimal impact on pressure rail readings
- Lockheed-Martin 1044 model will be used as test subject for new strut
  - Mach 1.1, 1.2, 1.3, 1.4, and 1.45
  - Angle of attack of -2, 2.1, and 6



STAR-CCM+ meshing

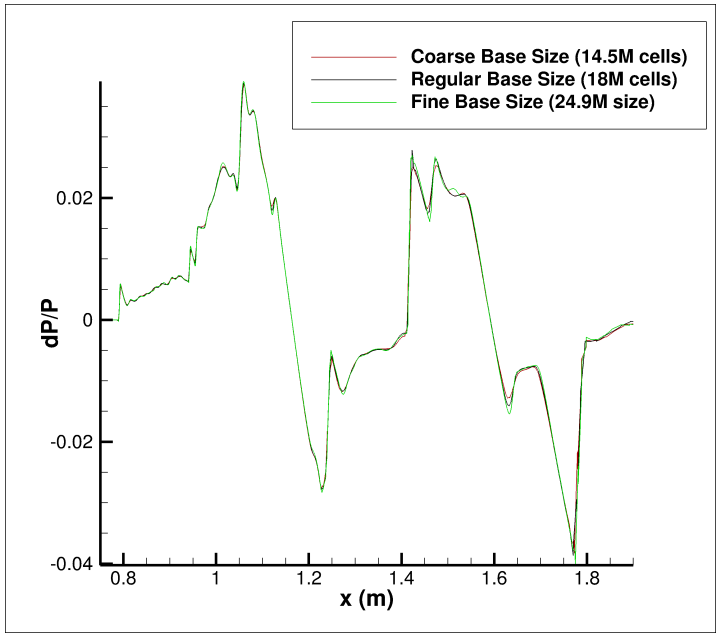
- LAVA group is providing unstructured CFD support for wind tunnel test
- STAR-CCM+ used for grid generation
  - Robust and quick to prepare
- Polyhedral cell geometry utilized for nearfield
- Farfield generated from extruded nearfield cell surfaces at proper Mach angle

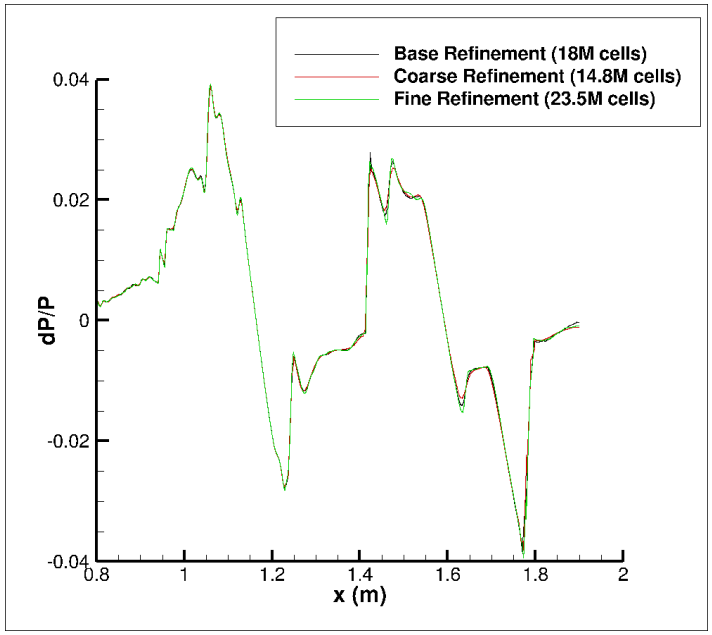




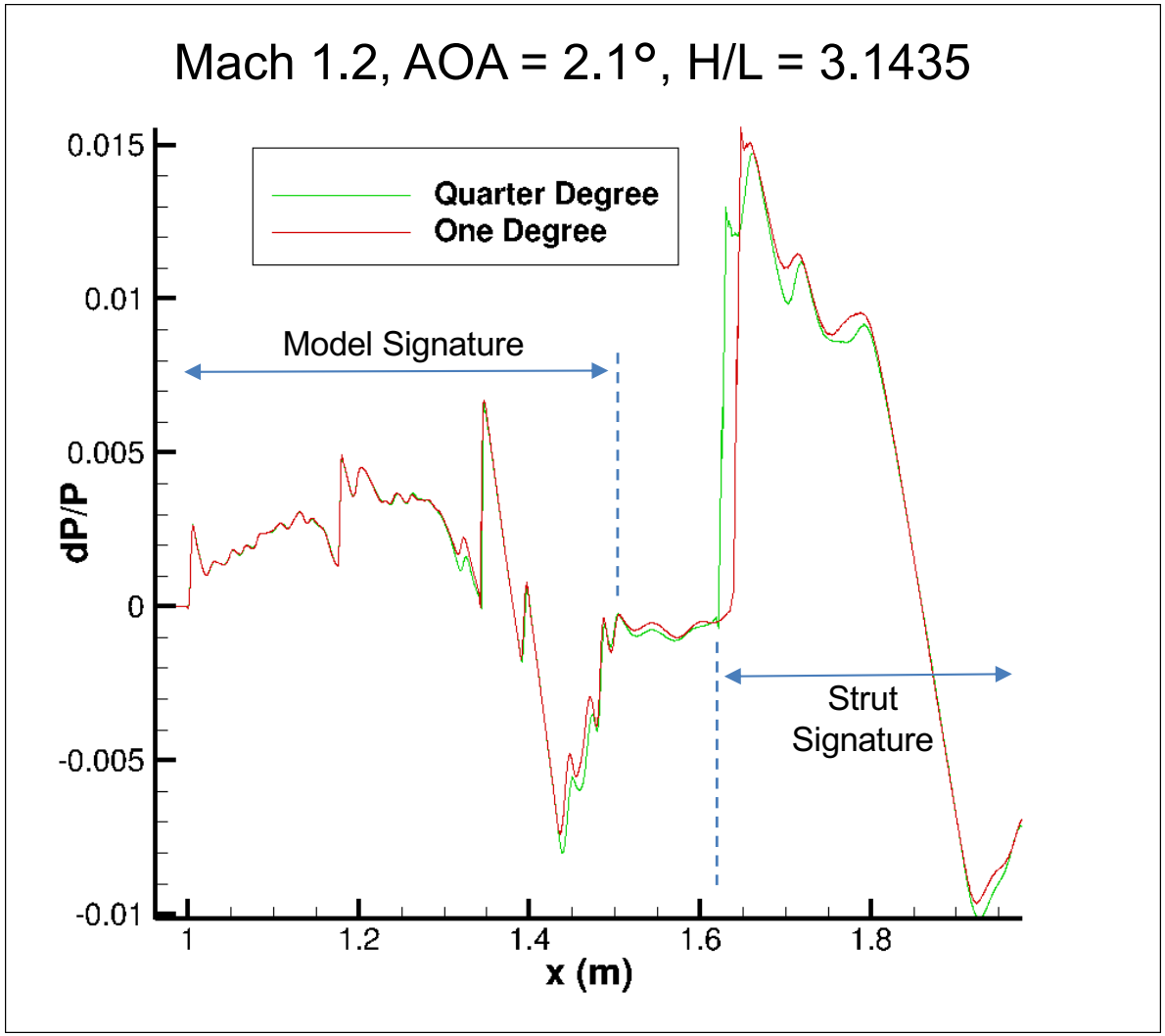
Refinement Study

- Performed refinement study for mesh sensitivity
- Adjusted overall base cell size and underbody refinement cell size
  - Increased and decreased base cell sizes
- Original base mesh appeared to have sufficient resolution
  - Pressure signature and loads of CST model were relatively unchanged
    - Mach 1.45, AOA = 2.1, H/L = 3.1435
  - Downstream signatures of less importance



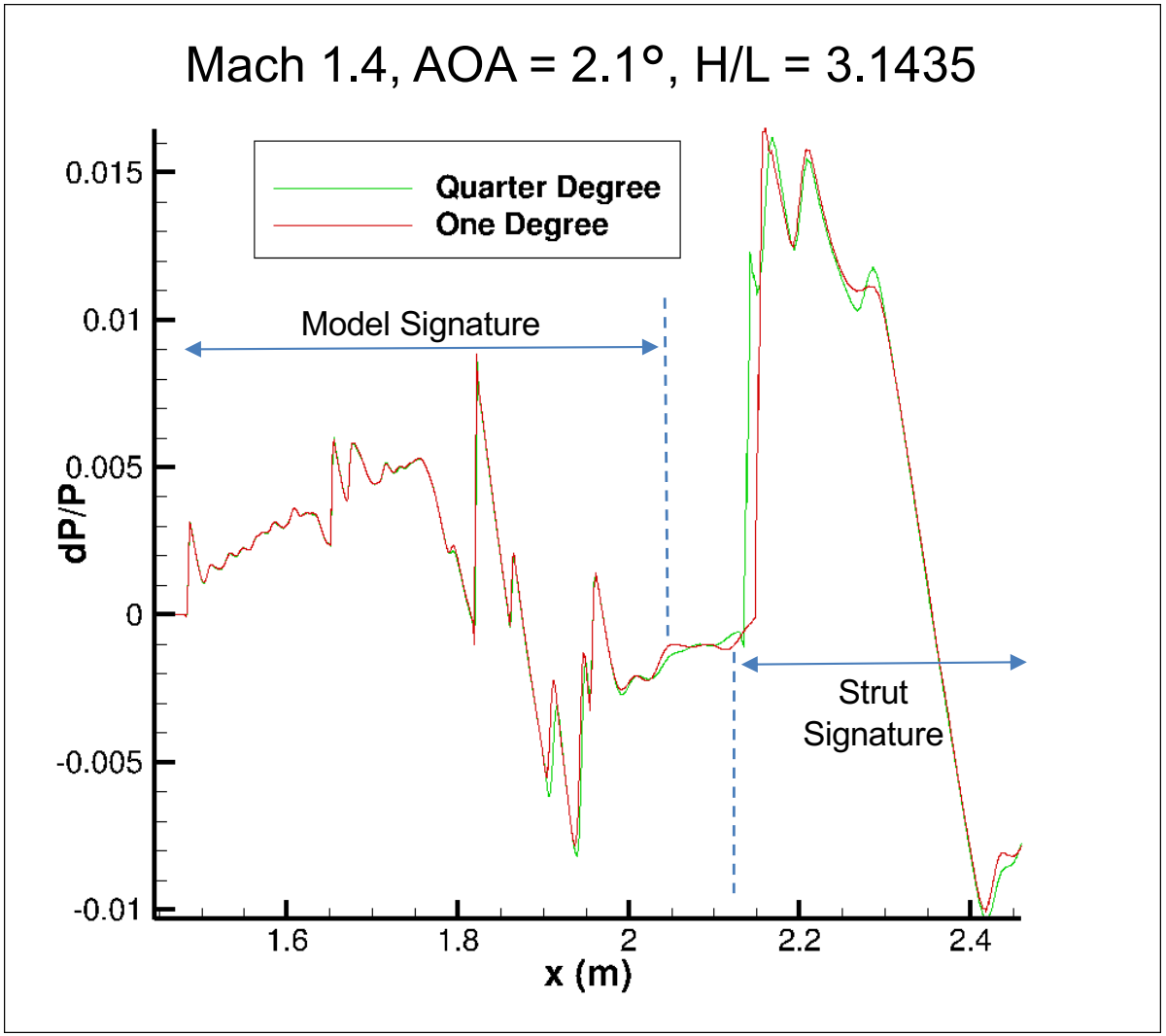


Results



- Latest results show very little impact on front part of signature
- Aft area shows slight deviation at lower Mach

Results



- Latest results show very little impact on front part of signature
- Aft area shows slight deviation at lower Mach

Internship Thoughts

Will Bowes

- Getting out of my comfort zone
- My group
- NASA resources

Bob Comstock

- Working with people of various expertise
-Application of new theories
-Vast software usage (LAVA, STAR-CCM+, ANSA, Pointwise, Bash and Tecplot)

- Future Improvement
  - Automation of mesh preparation process
  - Possibility of using pointwise as an unstructured grid generator
  - Developing more in-house scripts for key programs
  - Implementing best practices for new hires
  - One on one's with interns for more knowledge transfer

Thanks/Acknowledgements

- James Jensen - LAVA Research engineer/scientist
- Daniel Maldonado - LAVA Junior Research engineer/scientist
- Don Durston - Unitary Wind Tunnel Engineer
- Dr. Cetin Kiris - LAVA group lead and TNA branch chief